НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ім. ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки

Обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ Сергій СТІРЕНКО

“___”_____2020р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп’ютерні системи та мережі»

спеціальності 123 «Комп’ютерна інженерія»

на тему: «Система для аналізу загазованості повітря для Smart city»

Виконав: студент 4 курсу, групи ІО-63
Гайдай Анатолій Русланович _____ (підпис)

Керівник:
проф. д.т.н. Клименко Ірина Анатоліївна _____ (підпис)

Консультант з нормо-контроль:
проф. д.т.н. Сімоненко Валерій Павлович _____ (підпис)

Рецензент:
доц. к.т.н. Жданова Олена Григорівна _____ (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____ (підпис)

Київ - 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ім. ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки

Обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

"___"_____2020р.

ЗАВДАННЯ

на дипломний проєкт студенту

Гайдая Анатолія Руслановича

1. Тема проєкту «Система для аналізу загазованості повітря для Smart city», керівник проєкту Клименко Ірина Анатоліївна, проф., д.т.н., затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту 15 червня 2020 р.

3. Вихідні дані до проєкту: технічна документація, статистичні та теоретичні дані

4. Зміст пояснювальної записки опис предметної області, дослідження аналогів, дослідження технології створення системи, інструкція користувача.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) функціональна схема, принципова схема, структурна схема.

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормо-контроль	Сімоненко В. П. проф.		

7. Дата видачі завдання 01.09.2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми роботи	01.09.2019	
2.	Вивчення та аналіз завдання	15.10.2019	
3.	Розробка архітектури та загальної структури системи	16.11.2019	
4.	Розробка структур окремих підсистем	27.01.2020	
5.	Програмна реалізація системи	26.02.2020	
6.	Оформлення пояснювальної записки	11.03.2020	
7.	Захист програмного продукту	20.05.2020	
8.	Передзахист	26.05.2020	
9.	Захист	15.06.2020	

Студент Анатолій ГАЙДАЙ

Керівник Ірина КЛИМЕНКО

Анотація

В бакалаврському дипломному проєкті реалізована система для аналізу загазованості повітря для Smart city.

Система дозволяє вести спостереження за станом повітря та температури в місці встановлення пристрою. Програмний продукт складається з трьох частин, програмування пристрою на C, серверна частина з використанням Java та MySQL, мобільний додаток на Android(Java).

Для візуалізації, та налаштування використовується мобільний додаток на Android з використанням Google Map. В додатку, на карті відображаються маркери з інформацією про всі пристрої з останнім даними отриманими з пристроїв.

Annotation

The bachelor's degree project implements a system for analyzing air pollution for Smart city.

The system allows you to monitor the condition of the air and temperature at the installation site. The software product consists of three parts, programming the device in C, the server part using Java and MySQL, a mobile application on Android (Java).

For visualization and configuration, a mobile application on Android using Google Map is used. In the application, the map displays markers with information about all devices with the latest data received from the devices.

ОПИС АЛЬБОМУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4	ІАЛЦ.467200.002	Технічне завдання	3	
2	А4	ІАЛЦ.467200.003 ПЗ	Система для аналізу загазованості повітря для Smart city		
3			Пояснювальна записка	62	
4	А4	ІАЛЦ.467200.004	Функціональна схема	1	
5	А4	ІАЛЦ.467200.005	Принципова схема	1	
6	А4	ІАЛЦ.467200.006	Структурна схема	1	
7	А4	ІАЛЦ.467200.007	Лістинг програми	10	
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					

Змн.	Арк.	№ докум.	Підпис	Дата	Система для аналізу загазованості повітря для Smart city Відомість дипломного проєкту	Літ.	Арк.	Акрушів
Розроб.		Гайдай А. Р.					1	1
Перевір.		Клименко І.А.				КПІ ім. Ігоря Сікорського ФІОТ ІО-63		
Н. Контр.		Сімоненко В.П.						
Затверд.		Стіренко С.Г.						

ТЕХНІЧНЕ ЗАВДАННЯ

Технічне завдання

Зміст

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється для використання у курсі «Інтернет речей»

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського»

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи аналізу загазованості повітря для Smart city

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література по комп`ютерним системам, публікації в періодичних виданнях, довідники та публікації в Інтернеті щодо даної теми.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

Самостійність курсу – чіткі контури предмету вивчення.

Самодостатність – курс повинен містити необхідну інформацію та дані, які дозволяють в повній мірі розкрити мету курсу.

Коректність та актуальність інформації, яку охоплює даний курс.

5.2. Вимоги до програмного забезпечення

Операційна система LMint, MS Windows 7, MS Windows 8, MS Windows 10.

Java 8

MySQL

Android 7.1 та вище

5.3. Вимоги до апаратно частити

Комп'ютер на базі процесора Intel Pentium та вище

Оперативної пам'яті не менше 2048 Мбайт

Вільний простір жорсткого диску не менше 500 Мбайт

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	15.10.2019
Складання та узгодження технічного завдання	15.10.2019
Проектування програмного забезпечення	16.11.2019
Програмна реалізація продукту	26.02.2020
Тестування програмного забезпечення	27.02.2020
Налагодження та виправлення помилок	10.03.2020
Оформлення документації дипломної роботи	11.03.2020

Пояснювальна записка
до дипломного проєкту
на тему: «Система для аналізу загазованості повітря для Smart city»

Київ – 2020 р

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

TCP	Transmission Control Protocol Протоко́л керува́ння переда́чею
IP	Internet Protocol Інтернет протокол
SMTP	Simple Mail Transfer Protocol Простий Протокол Пересилання Пошти
GC	Graphics context Графический контекст
POP3	Post Office Protocol Поштовий Офісний Протокол
FTP	File Transfer Protocol Протокол передачі файлів
TFTP	Trivial File Transfer Protocol Тривіальний протокол передачі файлів
RARP	Reverse Address Resolution Protocol Зворотний протокол визначення адрес
ICMP	Internet Control Message Protocol міжмережевий протокол керуючих повідомлень
ARP	Address Resolution Protocol Протокол визначення адрес
RIP	Routing Information Protocol протокол маршрутизації
OSPF	Open Shortest Path First протокол динамічної маршрутизації
VDA	Video Distribution Amplifier Відео Підсилювач - розподільник
LSA	Latent semantic analysis Латентно-семантический анализ
EGP	Exterior Gateway Protocol протокол зовнішнього шлюзу
IETF	Internet Engineering Task Force відкрите міжнародне співтовариство проектувальників, учених, мережевих операторів і провайдерів, створене IAB в 1986 році, яке займається розвитком протоколів і архітектури Інтернету.
UDP	User Datagram Protocol Протокол датаграм користувача

GPL	General Public License Загальна публічна ліцензія GNU
AOT	Ahead-of-Time метод компіляції перед виконанням
JVM	Java Virtual Machine Віртуальна машина Java
API	Application Programming Interface Прикладни́й програ́мний інтерфе́йс
JIT	Just-in-time compilation компіляція «на льоту»

ВСТУП

Головним завданням даної роботи являється створення цілісної системи для аналізу загазованості повітря для Smart city. Дана система може бути використана на практиці, що б вести контроль за станом повітря у місті, а саме контроль рівня CO в повітрі.

В наш час проблема з якістю повітря у містах дуже серйозна, пожежі, природні катастрофи, збільшення кількості промислових об’єктів, транспорту призвела до того, що показники концентрації CO у місті знаходяться на небезпечному для людей рівні. Так велика кількість жителів міст отруюються чадним газом навіть не здогадуючись про це.

Отруєння можна поділити на три рівні. При першому рівні людина стає менш продуктивною, з’являються симптоми такі як головна біль, сухий кашель, запаморочення. При другому рівні сонливість та можливий параліч з збереженням свідомості. При третьому втрата свідомості та порушення дихання, що призводить до смерті. Для людей з хворобами дихальних шляхів, проблемами з кровоносною системою може бути смертельним навіть перший рівень отруєння. Якщо взяти до уваги, що кількість людей які мають схильність до набуття хвороби з цих категорій, або вже хворі, збільшується то проблема з CO в містах набуває більш серйозних наслідків що може дуже негативно вплинути як на економічний стан, через зменшення продуктивності працівників, так і на підвищення рівня смертності, яка стане прямим наслідком отруєння або смерть від симптомів викликаних отруєнням які стали причиною несчастних випадків, аварій. [7][8][9]

Оскільки даний газ не має запаху, кольору дізнатись про підвищення його рівня в приміщеннях без спеціального обладнання неможливо. Звичайні засоби індивідуального захисту не допомагають, а забезпечити спеціалізованими засобами всі підприємства не має можливості. В такому випадку потрібно мати систему яка буде моніторити стан повітря як на підприємствах так і в загальному в містах, це дозволить своєчасно запобігти

шкідливому вплину на здоров’я, також знайти причини підвищення рівня забруднення повітря.

При використанні системи з великою кількістю датчиків які будуть формувати карту стану повітря на території всього міста можна буде знайти джерело забруднення та швидко ліквідувати джерело. При використанні системи моніторингу разом з камерами відеоспостереження можна підвищити ефективність пожежних бригад, за рахунок своєчасного отримання інформації про підвищення рівня CO в повітрі та отримання відео з місця можливого займання. Так можна знайти джерела займання як на території підприємств, житлових секторів так і на територіях які закриті для вільного відвідування та на яких перебування співробітників можливе тільки у відведений час.

За рахунок збільшення покриття територій безперервним інтернет з’єднанням можливо ввести систему моніторингу вмісту CO на значній площі, покривши не тільки міста і підприємства, а також автомагістралі, залізницю. Це в свою чергу допоможе контролювати якість повітря на території яка забезпечена засобами моніторингу, швидку реакцію на виникнення загроз погіршення якості повітря та загальне підвищення рівня екологічного контролю.

Враховуючи все вище зазначене, метою даної роботи було створення системи яка зможе збирати дані по вміст CO в повітрі, передавати їх на сервер, та у зручному вигляді відображати отримані дані у зручному для користувачів вигляді. Додатково було додано вимір температурних показників у місці розташування пристрою шляхом додання цифрового датчику температури до схеми. Так було отримано пристрій на базі плати Nodemcu з використанням датчику газу MQ - 7 та датчику температури 18b20. Для відображення даних використовується додаток на базі Android.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ АНАЛІЗУ ПОВІТРЯ В МІСТАХ

1.1. Актуальність.

Завдання даної роботи складається з створення пристрою для моніторингу вмісту CO та температури повітря, створення серверної частини яка буде взаємодіяти з базою даних, створення мобільного додатку для зручного відображення даних користувачам системи. Після програмування пристрою користувач має можливість налаштувати його вказавши унікальний ідентифікаційний код пристрою в мобільному додатку та передавши йому координати розташування.

Перша частина теми система аналізу загазованості повітря для Smart city це проектування апаратного пристрою та його програмування відповідно до мети роботи, сюди входить вибір обладнання необхідного для створення пристрою, а також написання програми збору даних з датчиків та їх подальша передача на сервер.

Друга частина це створення серверної частини яка буде взаємодіяти з базою даних та передавати дані між пристроями аналізу та мобільними додатками користувачів.

Третя частина являє собою мобільний додаток, з використанням карти google map, на якому відображаються маркери з інформацією про стан повітря та температури згідно з налаштуваннями розміщення пристроїв.

1.2. Аналіз існуючих рішень.

Існує багато сервісів які надають послуги з аналізу стану повітря в містах. Починаючи з звичайних сервісів прогнозу погоди і закінчуючи спеціалізованими сервісами з комерційною складовою.

Одним із таких сервісів є Kunak. Даний сервіс надає змогу вести спостереження за станом повітря у місті або на виробництві з можливістю відображати дані у зручному вигляді, використовуючи таблиці, графіки. Але

даний сервіс є комерційним продуктом і для використання потрібно заключити контракт з компанією виробником. [1]

Ще одним сервісом з послугами моніторингу є IQAir який для відображення даних використовує карту світу яка крім рівня забруднення відображає рух повітряних потоків. Хоч компанія і надає послуги з моніторингу її основна спеціалізація це засоби очищення повітря для офісів, транспорту, житлових секторів. [3]

Сервіс Aeroqual теж дає можливість моніторингу, але також після заключення контракту з фірмою виробником.[2]

Хоч дані сервіси і надають можливість вести спостереження за станом повітря, але вони мають свої недоліки такі як мала площа покриття у IQAir, оскільки сервіс має пристрої моніторингу тільки у невеликій кількості та у великих містах, оскільки дані отримують з загальних джерел таких як академій та інститутів, або інших наукових-дослідницьких установ, та пристроїв індивідуального аналізу. Також потрібно брати до уваги що обладнання платних сервісів має достатньо великий об’єм і його установка може бути проблематичною через брак простору для монтування.

Розглянемо більш детально існуючі рішення для аналізу стану повітря.

1.2.1. Сервіс IQAir.

Головна сторінка веб сайту сервісу IQAir на рис. 1.1. [3]

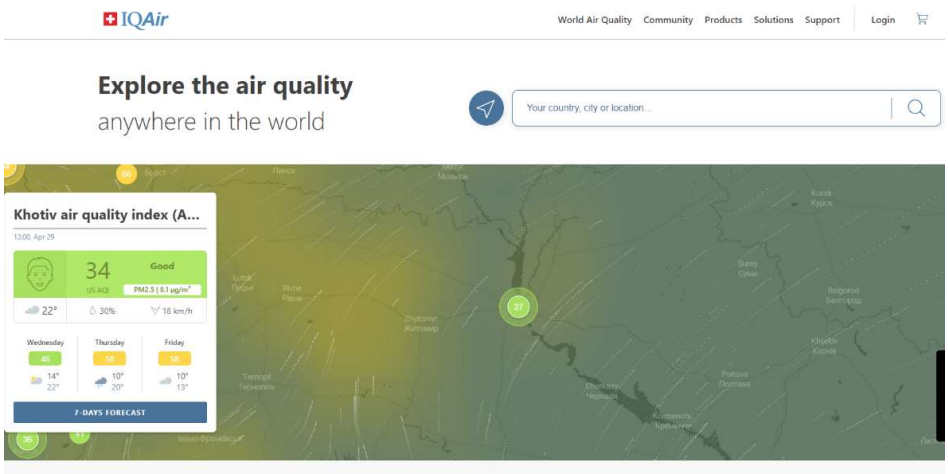


Рис. 1.1. Головна сторінка веб сайту сервісу IQAir. [3]

На головній сторінці відображається мапа з кольоровими зонами відповідно до якості повітря, поточний рівень у вибраному місті у додатковому вікні та на карті. Наявний пошук за містом. Також наявна таблиця з найгіршим станом повітря за містами та країнами (рис. 1.2).

Air quality and pollution city ranking

29 April 2020, 15:39

	Major city	US AQI	Followers	
1	Beijing, China	162	5.94M	Follow
2	Shenyang, China	154	36.9K	Follow
3	Hanoi, Vietnam	144	3.47M	Follow
4	Chengdu, China	139	1.78M	Follow
5	Hangzhou, China	127	123K	Follow
6	Riyadh, Saudi Arabia	120	37.3K	Follow
7	Jakarta, Indonesia	116	2.14M	Follow
8	Yangon, Myanmar	105	59.4K	Follow
9	Phnom Penh, Cambodia	98	204K	Follow
10	Dhaka, Bangladesh	97	117K	Follow
11	Chongqing, China	97	71.7K	Follow

Рис. 1.2. Таблиця міст з найгіршим повітрям. [3]

Таблиця відображає рівень забруднення, кількість людей які стежать за конкретним містом.

На інтерактивній карті можна знайти найближче місце аналізу та поточний результат. Внизу сторінки є кольоровий покажчик якості(рис. 1.3). Даний сервіс має мобільний додаток який надає можливість придбати засоби аналізу та відображає поточні дані з індивідуальних засобів інших користувачів сервісу(рис. 1.4).

Рис. 1.3. Інтерактивна карта з поточними даними. [3]

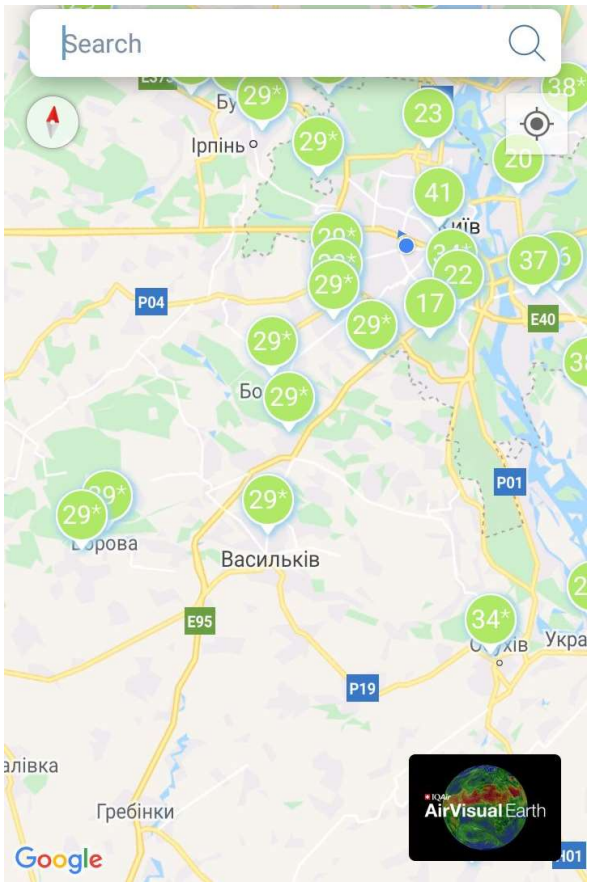


Рис. 1.4. Інтерактивна карта в мобільному додатку. [3]

1.2.2. Сервіс Kunak.

Головна сторінка веб сайту сервісу Kunak на рис. 1.5. [1]

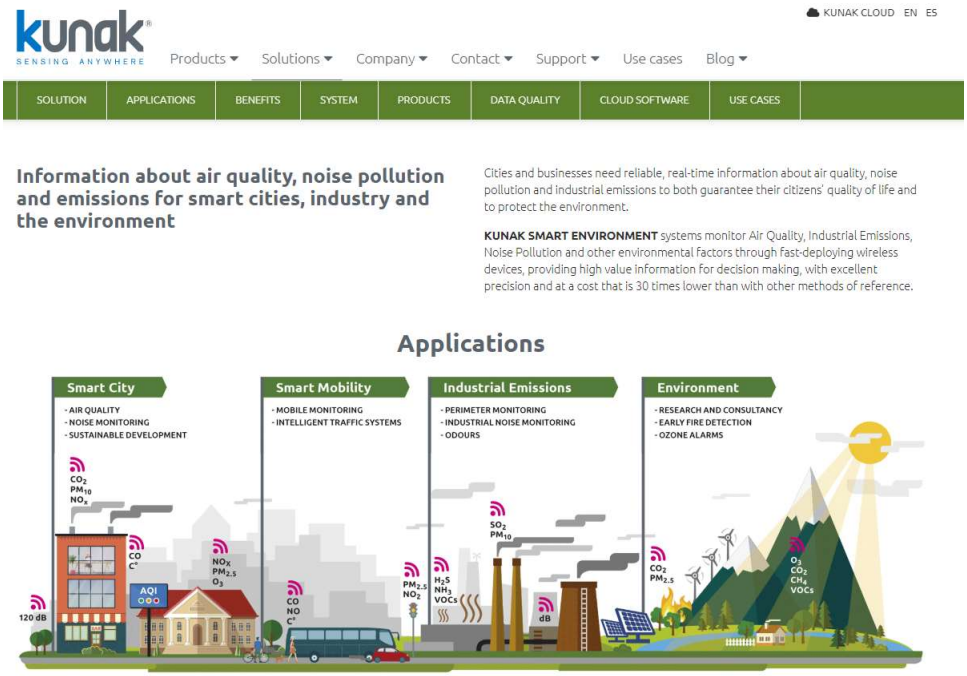


Рис. 1.5. Головна сторінка веб сайту сервісу Kunak. [1]

На головній сторінці даного сервісу показано спеціалізацію обладнання даної компанії для аналізу в залежності від місця проведення аналізу. Далі на головній сторінці є загальна схема системи(рис. 1.6).

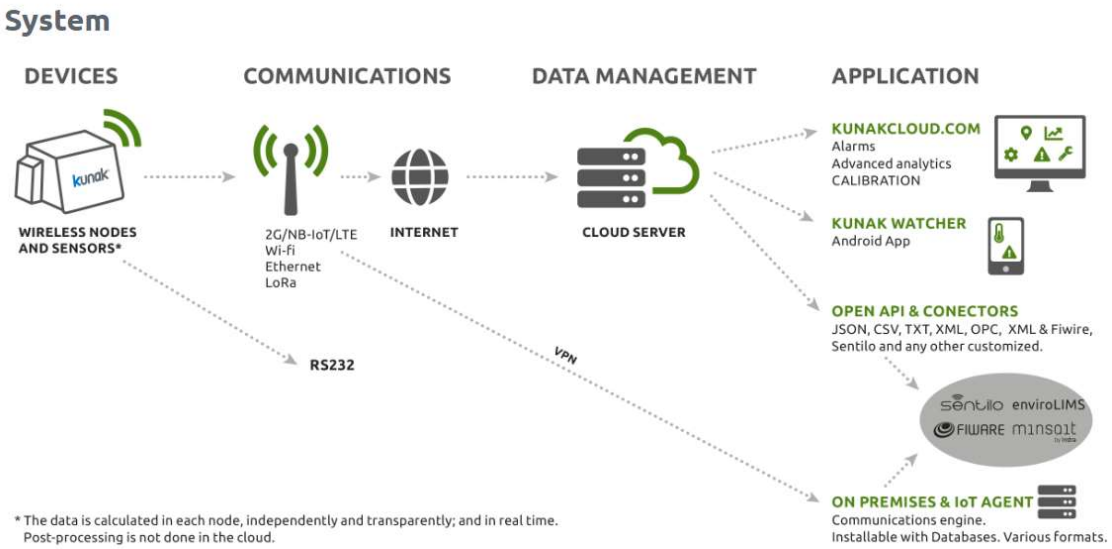


Рис. 1.6. Схема системи сервісу Kunak. [1]

Оскільки сервіс працює виключно в платному режимі то наявна інформація тільки з демонстраційних слайдів відносно відображення

даних аналізу. З карти можна отримати інформацію з маркерів які відображають розташування пристроїв аналізу(рис. 1.7). Графіки аналізу на часовій діаграмі(рис. 1.8).

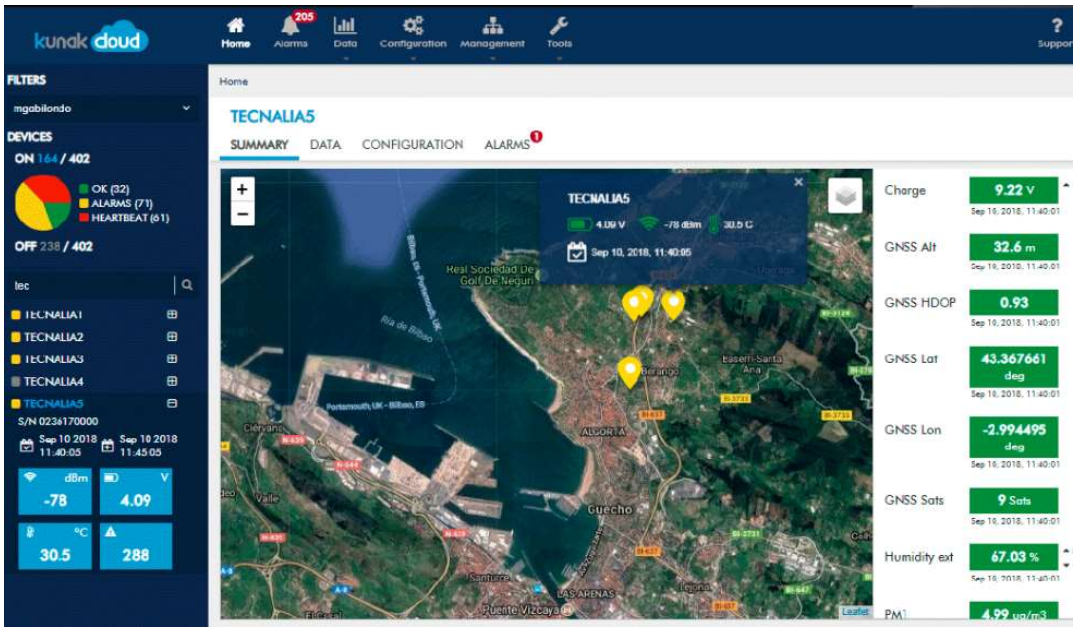


Рис. 1.7. Карта розташування пристроїв. [1]

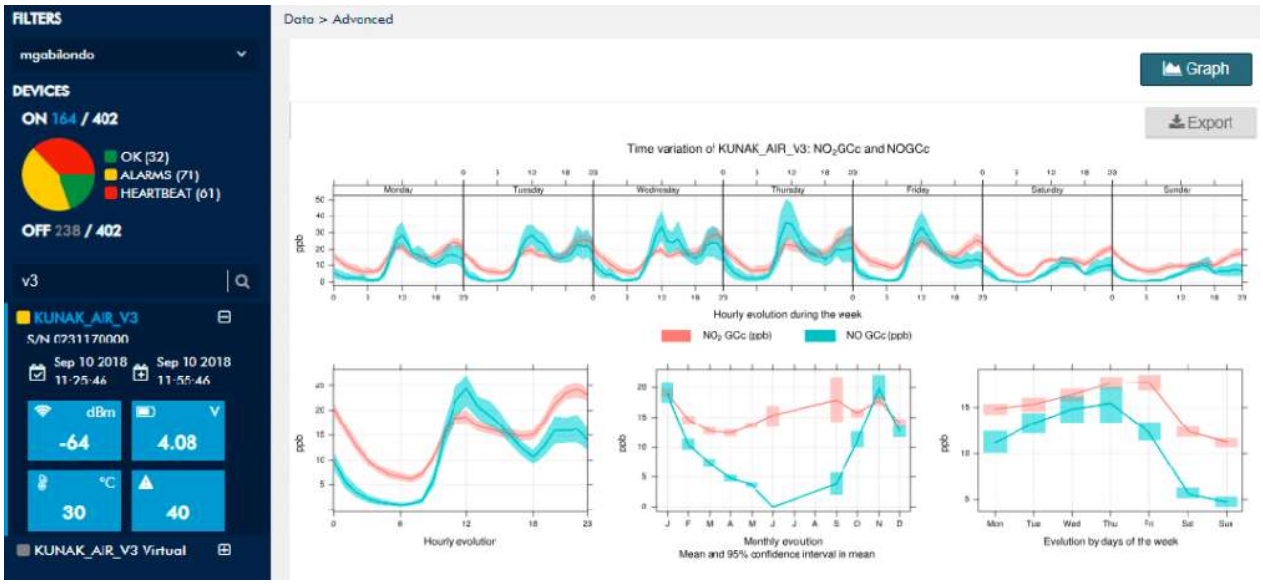


Рис. 1.8. Часова діаграма вмісту газу в повітрі. [1]

Обладнання розташовується в металічному корпусі та може містити засоби живлення у вигляді сонячних батарей(рис. 1.9).

Рис. 1.9. Пристрій аналізу повітря з живленням. [1]

1.2.3. Сервіс Aeroqual.

Головна сторінка веб сайту сервісу Aeroqual на рис. 1.10. [2]

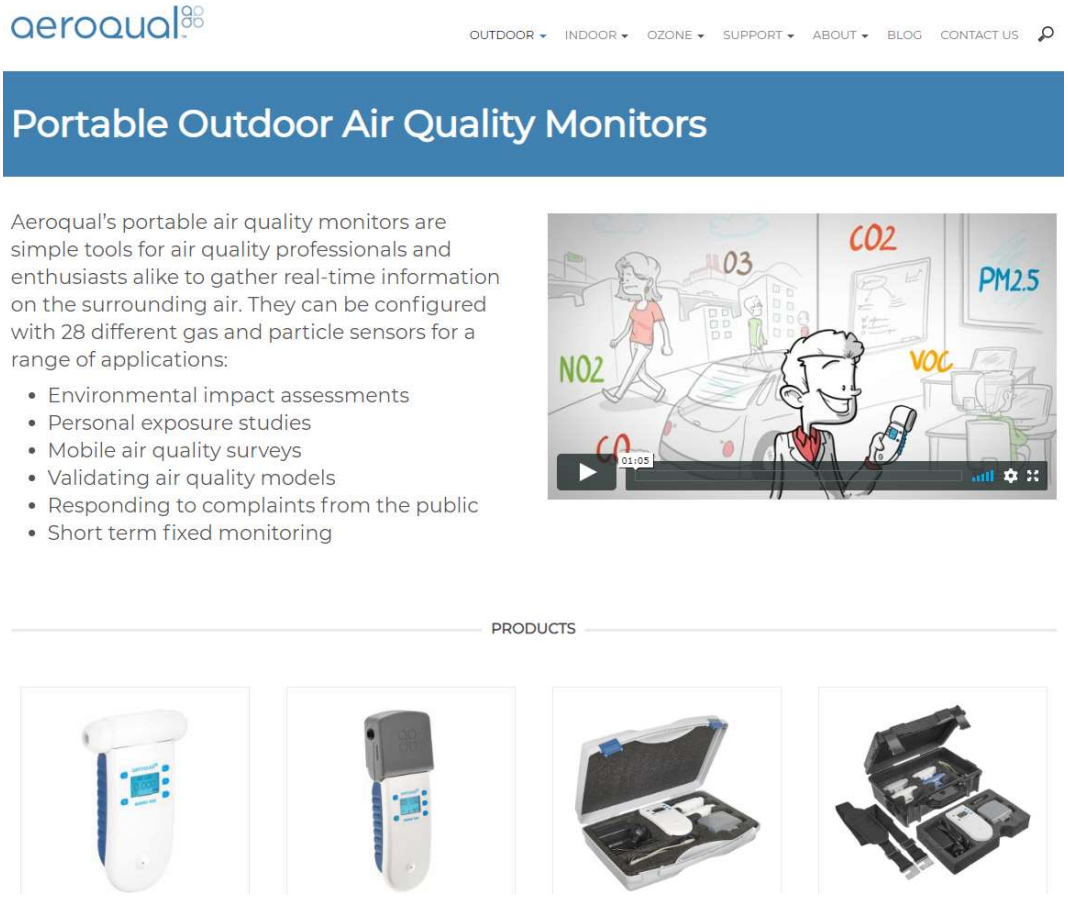


Рис. 1.10. Головна сторінка веб сайту сервісу Aeroqual. [2]

Головна сторінка містить інформацію чому потрібен даний сервіс та пристрої які використовуються для аналізу. Пристрій аналізу в розрізі(рис. 1.11).

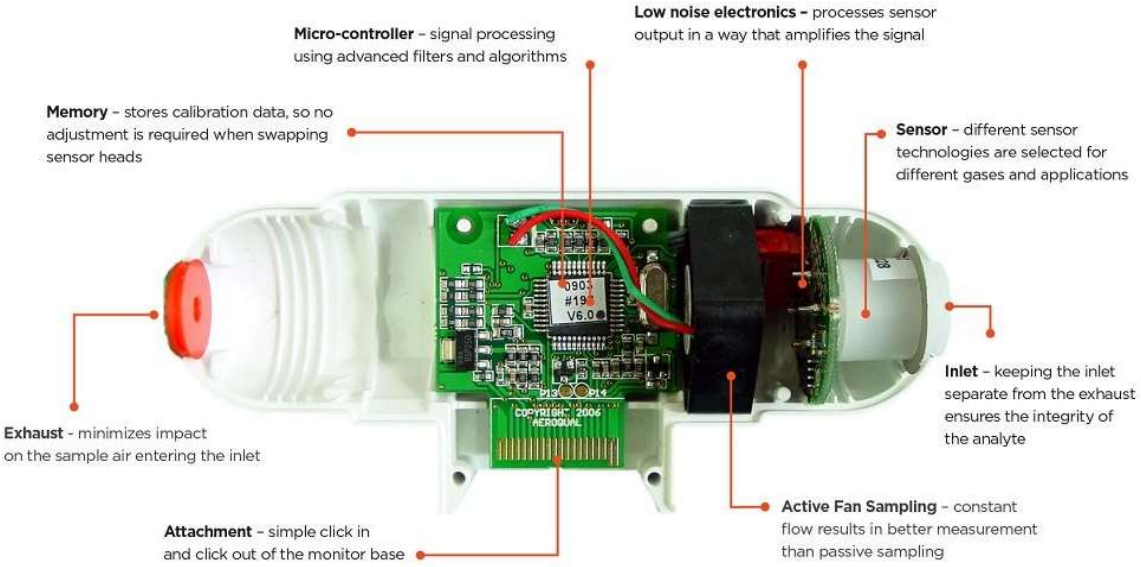


Рис. 1.11. Пристрій аналізу в розрізі. [2]

Результати аналізу можна отримати на дисплей пристрою або підключити до системи і отримати у вигляді часової діаграми або у вигляді таблиці(рис. 1.12).

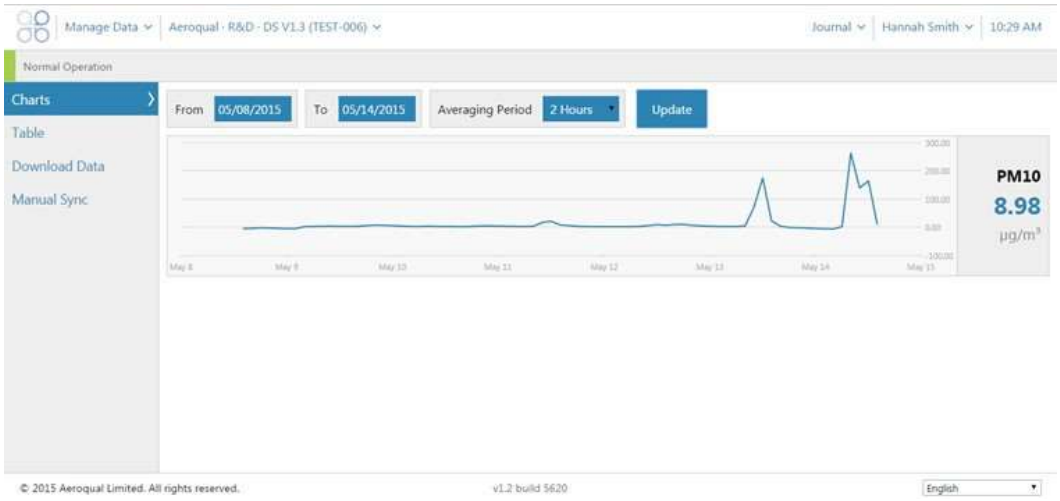


Рис. 1.12. Часова діаграма результатів аналізу повітря. [2]

Висновки до розділу 1. Постановка задачі

В даному розділі було розглянуто існуючі рішення аналізу загазованості повітря. Розглянуті сервіси в загальному виконують поставлену задачу але у кожного з них є свої недоліки.

IQAir надає якісне відображення даних на карті як через веб сайт так і на мобільному додатку але тільки у мобільному додатку вказані переносні пристрої аналізу. Також мала кількість станцій та пристроїв аналізу не дозволяє отримати повну картину результатів, а лише у густо населених містах.

Kunak хоч і надає можливість отримання результатів аналізу у різних формах але потребує платної підписки та фірмового обладнання для роботи, а за рахунок вартості та кількості пристроїв аналізу які можна отримати ми не можемо покрити значну територію.

Aeroqual в основному використовує портативні пристрої аналізу, що не дозволяє отримати повну і точну картину стану повітря, оскільки показники пристрою можуть бути спотворені не правильним використанням або іншими чинниками.

Тому основною задачею дипломного проекту було створення власної системи яка позбавиться недоліків вище зазначених сервісів та буде мати переваги у простоті використання, низькій вартості, швидкому налаштуванні, швидкому ремонту та заміні, та враховуючи кліматичні особливості регіону вибране обладнання зможе працювати тривалий час без збоїв.

РОЗДІЛ 2
ОГЛЯД ОБЛАДНАННЯ ТА АНАЛІЗ ДОСТУПНИХ ЗАСОБІВ ПРОЕКТУВАННЯ

2.1. Сокет

Сокет вказує на програмне абстрагування, призначене для використання стандартних та спільних API для передачі та прийому даних через мережу або механізм IPC. Це момент, коли прикладний код процесу здійснює доступ до каналу зв'язку через з’єднання, отримуючи зв'язок між процесами, які працюють на двох фізично окремих машинах. З точки зору програміста, сокет - це певний об'єкт, на якому можна зчитувати та записувати дані, що передаються чи приймаються. [15]

Сокети були представлені в 1983 році в BSD і потім були перейняті практично всіма іншими операційними системами. З цієї причини функції програмування сокетів зазвичай називають API сокета Berkeley.

Вони дозволяють встановлювати різні комунікації всередині мережі, використовуючи стек TCP / IP. В основному вони складаються з двох частин:

IP-адреса: унікальний код відповідної машини, розмір 32 біт.

Номер порту: номер, розміром 16 біт, який використовується єдиним процесом для запиту протоколу всередині машини.

Ця комбінація дозволяє однозначно відрізняти окремі запити всередині мережі.

Сокети, які використовують одні і ті ж протоколи, укладені в одній сім'ї, що називається доменом. В основному існує два типи доменів:

Інтернет-сокет: дозволяє передавати дані між процесами, розміщеними всередині підключених віддалених машин, наприклад, що належать одній і тій же локальній мережі.

Сокет Unix: дозволяє передавати дані між процесами, що належать до однієї машини.

Сокети в основному класифікуються на три категорії, кожна характеризується власним режимом зв'язку.

Datagram Socket: цей тип сокета використовує з'єднання на основі протоколу UDP. Це означає, що надсилання даних відбувається за допомогою передачі невеликих дейтаграм, не гарантуючи правильного порядку надходження та правильності інформації. Клієнт і сервер не встановлюють реального зв'язку, але клієнт спілкується безпосередньо з сервером, коли він захоче.

Stream Socket: вони використовують з'єднання на основі протоколу TCP, отже, орієнтоване на з'єднання та надійне.

З'єднання встановлюється за допомогою наступної послідовності:

Клієнтські та серверні процеси ініціалізуються.

Клієнтський процес надсилає запит на з'єднання на сервер із зазначенням сокета.

Сервер приймає запит і створює віртуальний канал, який він використовуватиме для передачі даних.

Raw Socket: цей тип розеток використовується для розробки конкретних протоколів.

У свою чергу в IP є два типи сокетів:

Слухаючі сокети, які представляють можливість отримання нових з'єднань. Сокети такого типу ідентифікується транспортним протоколом, IP-адресою комп'ютера, номером порту;

Сокетні пари, які являють собою певний активний зв’язок. Сокети такого типу ідентифікується транспортним протоколом, використовує чотири складові в пакеті: IP-адресу джерела, IP-адресу пункту призначення, номер порту джерела, номер порту призначення.

2.2. TCP/IP

Набір протоколів Інтернету в IT та телекомунікаціях вказує на сімейство мережевих протоколів, пов'язаних залежностями використання, на яких базується логічне функціонування Інтернету. Набір протоколів називають

TCP/IP відповідно до двох найважливіших протоколів, визначених у ньому: протоколу управління передачею (TCP) та Інтернет протоколу (IP), відповідної шарованої моделі архітектури мережі. Він представляє фактичний стандарт у галузі мереж передачі даних на відміну від стандарту, представленого моделлю ISO / OSI. [4]

На початку 1970-х Агенція прогресивних дослідницьких проектів оборони фінансувала університети Стенфорда та BBN (Болт, Беранек і Ньюмен) на розробку набору протоколів зв'язку, які будуть використовуватися для розвитку мереж. Пакет комутований, для взаємозв'язку комп'ютерів. Так народився пакет Internet Protocol Suite, два найвідоміші протоколи якого TCP та IP.

Протокол IP забезпечує систему адресації вузлів мережевого терміналу, присвоюючи кожному унікальне ім'я, що складається з чотирьох груп цифр. На наступному більш високому рівні протокол TCP управляє потоком інформації між двома вузлами. Ця мережева архітектура називається абревіатурою TCP / IP або IP / TCP. Творцями цих протоколів передачі, які все ще використовуються в Інтернеті, є конкретно Роберт Кан і Вінтон Серф, яким колишній президент США Джордж Буш вручив президентський медаль свободи, що є найвищою з відзнак зірок та смуг цивільних осіб, 9 листопада 2005 р. Два науковці не новачки в цьому виді нагород: на початку 2005 року їм було присвоєно престижну 2004 р. Премія Тьюрінга, еквівалент Нобелівської премії в галузі інформаційних технологій.

Серф і Кан розробили стандарт передачі пакетів через мережу в 1973 році, під час роботи над проектом розвитку систем зв'язку. В даний час Вінтон Серф співпрацює з Google, щоб створити стандарти для майбутніх додатків, а тим часом він присвячений розробці нових протоколів міжпланетної комунікації для лабораторії реактивного руху NASA. Однак Роберт Кан після 13 років служби в DARPA став президентом Корпорації національних дослідницьких ініціатив.

Протоколи знаходяться у вільному доступі з часів своєї розробки. Це створює певні неоднозначності через те, що правильною назвою буде Internet Protocol Suite. Наприклад, трапляється чути про послуги на базі TCP / IP навіть тоді, коли насправді замість TCP використовується альтернативний протокол UDP, який також належить до пакету Internet Protocol Suite. Як правило, TCP використовується для тих додатків, які потребують послуги, орієнтованої на з'єднання, наприклад, обміну електронною поштою та файлами, тоді як UDP все частіше сприймає додатки в режимі реального часу, такі як онлайн-ігри або потокове передавання аудіо та відео; відмінність між двома протоколами полягає у більшій надійності при транспортуванні даних TCP, яка пропонує низку спеціально розроблених сервісів (управління потоком, перевантаження), в той час як UDP значною мірою покладається на швидкість передачі за рахунок надійності. Тому завжди майте на увазі, що абревіатура TCP / IP настільки поширена, що використовується іноді, навіть якщо є більш правильні альтернативні терміни.

Архітектура, прийнята Інтернетом, - це TCP / IP, яка, зарекомендувала себе як фактичний стандарт завдяки розповсюдженню та масовим інвестиціям, пов'язаним з Інтернетом.

Цей набір можна описати за аналогією з моделлю OSI, яка описує шари стеку протоколів. У групі протоколів кожен рівень вирішує низку проблем щодо передачі даних і надає чітко визначену послугу на найвищих рівнях. Вищі рівні логічно наближаються до користувача та працюють з більш абстрактними даними, залишаючи завдання нижчих рівнів перекладати дані у форми, за допомогою яких ними можна фізично маніпулювати та остаточно передавати по каналу зв'язку.

Інтернет-модель була розроблена як рішення практичної інженерної проблеми, оскільки необхідно було поступово додавати протокольні шари до мережевої архітектури локальних мереж, щоб отримати ефективний та надійний взаємозв'язок. В іншому сенсі модель OSI була більш теоретично-дедуктивним підходом і також вироблялася в більш старій мережевій моделі.

2.2.1. Прикладний рівень

Перший рівень - це додаток: він представляє інтерфейс з користувачем і дозволяє, наприклад, конструювати веб-сторінки, встановлювати та керувати робочими сеансами клієнтських процесів між нашим браузером та веб-сервером.

Керує повідомленнями створеними клієнтами та серверами, і передає їх у вигляді пакетів через повно дуплексне з'єднання; успішне завершення відправлення підтверджується квитанцією про повернення або підтвердження. Це також віртуальний зв’язок між двома програмами, деталі яких делеговані на наступний рівень, який називається транспортом.

Серед традиційних послуг, які надають протоколи прикладного рівня з сімейства TCP / IP, найбільш популярними сьогодні є електронна пошта - SMTP і POP3, FTP та TFTP – для передачі файлів, TELNET – для емуляції віддаленого терміналу і інші.

2.2.2. Транспортний рівень

Транспортний рівень пропонує послугу на рівні додатків, використовуючи послуги базового мережевого рівня.

Для управління кількома процесами, активними в передачі даних на одному вузлі (або на комп'ютері), тобто декількох сеансах активного перегляду, транспортний рівень (TCP або UDP) використовує кілька номерів портів.

TCP при відправці пакетів використовує механізм "Розсувне вікно". TCP надсилає серію пакетів відповідно до конкретних правил:

- У кожному надісланому вікні пакета передавач запускає тайм-аут;
- Одержувач надсилає ACK для кожного прийнятого пакету із зазначенням наступного очікуваного пакету;
- Тому передавач вважає всі попередні пакети відвантаженими;

- Якщо час закінчується або надходять 3 повторюваних ACK, TCP припускає, що один або більше пакетів втрачено, і вживає заходів для повторної передачі.

Це дуже важлива методика, оскільки вона забезпечує надійний канал зв'язку. Крім того, TCP містить механізми для управління заторами та управління потоком.

2.2.3. Міжмережевий рівень

Інтернет-протокол (IP) - це протокол моделі DARPA (за моделлю OSI він класифікований на мережевому рівні), який управляє адресацією вузлів та маршрутизацією: кожному вузлу фактично присвоюється IP-адреса яка визначить його однозначно в мережі; Особливості маршрутизації, з іншого боку, дозволяють вибрати найкращий шлях для передачі повідомлення до певного вузла одержувача, відомий як його IP-адреса.

2.2.4. Канальний рівень

Канальний рівень знаходиться внизу еталонної модель TCP / IP він вказує, що повинен бути рівень доступу до мережі, здатної надсилати IP-пакети. У моделі ISO / OSI цей шар відповідає першим двом рівням: шару зв'язку та фізичному шару.

На рівні з'єднання ви вирішуєте, як перенести повідомлення для кожного окремого ділянки маршруту: від комп'ютерного веб-браузера до першого маршрутизатора, від першого маршрутизатора до другого, від другого до третього і так далі. Це віртуальний зв’язок між двома сусідніми комп'ютерами (або маршрутизаторами). Також у цьому випадку комунікаційні інтерфейси сусідніх вузлів будуть ідентифіковані за допомогою унікальної адреси, яка зазвичай називається MAC-адресою.

Фізичний рівень передає повідомлення по каналу зв'язку, як правило, у вигляді електричних або електромагнітних сигналів, хоча можливе також використання акустичних хвиль (наприклад, у підводних сенсорних мережах).

2.3. NodeMcu v3

NodeMcu(рис. 2.1.) – платформа на основі ESP8266 для створення пристроїв інтернету речей (IoT). Модуль може отримувати і відправляти дані в локальну мережу або інтернет за допомогою Wi-Fi, використовувати для одно сторінкового невеликого сайту. [12]

Рис. 2.1. NodeMcu v3 [12]

2.3.1.Технічні характеристики модуля:

- Підтримка Wi-Fi протокол 802.11 b/g/n;
- Підтримка режиму точка доступа, клієнт;
- Вхідна напруга 3,7В – 20 В;
- Робоча напруга 3В-3,6В;
- Максимальний струм 220мА;
- Вбудований стек TCP/IP;
- Робоча температура від – 40 ° С до + 125 ° С
- 80 МГц, 32-бітний процесор;
- Час пробудження та відправки пакетів 22мс.

Даний модуль за замовченням можна програмувати на мові Lua, але він може бути запрограмований на мову С для Arduino. Для роботи з платою використовують Arduino IDE(рис. 2.2).

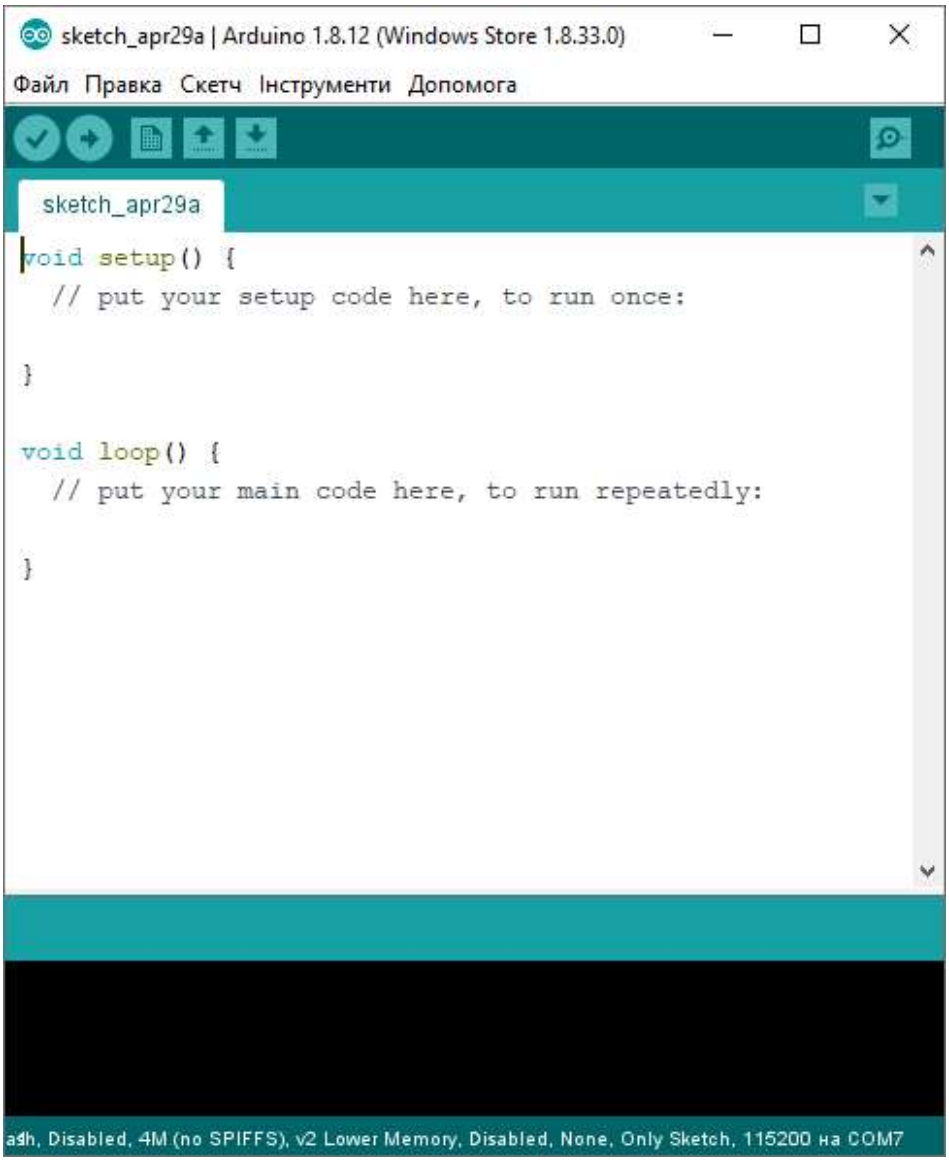


Рис. 2.2. Arduino IDE

Дана програма проста у використані, легка у встановлені. Для роботи з сімейством плат NodeMcu потрібно підключити репозиторій для цього сімейства через налаштування. Це збільшить кількість плат які можна програмувати, додаючи нові налаштування для плат. Оскільки плати та вся технічна документація по їх розробці та виготовленню поширюється в

вільному доступі детальні налаштування для модуля дозволять уникнути помилок програмування. [14]

Бібліотеки для роботи можна завантажити через програму, більшість з них розташовані на github. Також основна частина бібліотек містить приклади для конкретних плат, що полегшує роботу та навчання.

2.4. MQ-7 – датчик газу

MQ – 7 Датчик газу CO. Конструкція датчика представлена на рис. 2.3. [6]

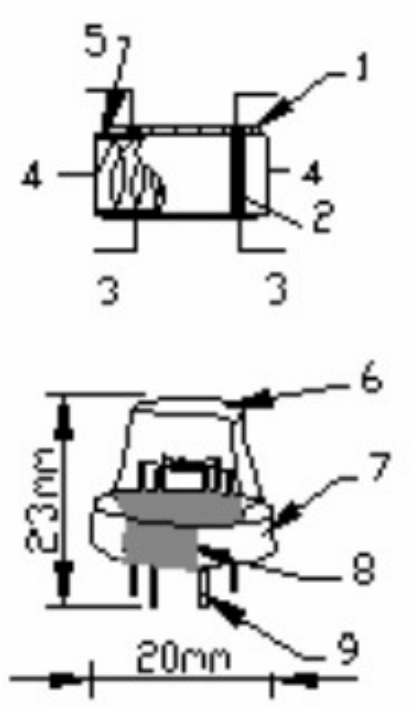


Рис. 2.3. Конструкція датчика газу:

1- Газо чутливий шар;

2- Електрод;

3- Електродна лінія;

4- Котушка нагріву;

5- Керамічна трубка;

6- Анти вибухова сітка;

7- Кільце кріплення;

8- Смоляна основа;

9- Ніжка. [6]

2.4.1.Технічні характеристики датчика:

- Живлення в діапазоні від 3.3 до 5 В;
- Вимірює вміст CO від 20ppm до 2000 ppm. ;
- Робоча температура від –20 ° C до + 50 ° C;
- Похибка: ± 5 ppm.

2.4.2. Принцип роботи датчика.

Датчик MQ-7 відноситься до напівпровідникових приладів. Принцип роботи датчика бере за основу зміну опору тонко плівкового шару діоксиду олова SnO2 при контакті з молекулами газу. Діоксид олова та керамічна трубка з покриттям Al2O3 являє собою чутливий елемент датчика. Всередині трубки розміщений нагрівальний елемент, який підвищує температуру чутливого шару до температури, при якій він починає реагувати на визначений газ. Чутливість до різних газів досягається різницею у елементному скалі чутливого шару.

2.5. DS18B20 – датчик температури

DS18B20 – цифровий датчик температури (рис. 2.4). [5]

2.5.1.Технічні характеристики датчика:

- Кожен пристрій має унікальний 64-розрядний серійний код зберігається у вбудованому ПЗУ;
- Живлення в діапазоні від 3 до 5.5 В;
- Вимірює температуру від –55 ° C до + 125 ° C;
- Похибка: ± 0,5 ° C точність від –10 ° C до + 85 ° C;
- Роздільна здатність термометра вибирається користувачем від 9 до 12 біт;
- Перетворює температуру в 12-бітове цифрове слово за 750 мс;
- Доступний у 8-контактному SO, μSOP, та 3-контактні пакети TO-92;

Кожен DS18B20 має унікальний 64-розрядний серійний код, який дозволяє декільком DS18B20 функціонувати на одній шині; таким чином, просто

використовувати один мікропроцесор для управління багатьма DS18B20, розподіленими по великій площі.

2.5.2. Принцип роботи датчика.

На рис. 2.6. Показана схема підключення датчика до плати. Для коректної роботи датчик потрібно підключити резистор 4700 ОМ між ніжкою живлення та ніжкою даних.

Обмін інформацією з датчиком відбувається завдяки таким операціям:

- Ініціалізація - визначення послідовності сигналів, з яких починається вимір і інші операції. Мікропроцесорний пристрій подає імпульс скидання, після цього датчик повинен подати імпульс присутності, повідомити про готовність до виконання операції;
- Запис даних - відбувається передача байт даних в датчик;
- Читання даних - відбувається прийом байт даних з датчика.

Рис. 2.4. Конструкція датчика температури:

1 - Земля;

2 – Пін даних;

3 - Живлення. [5]

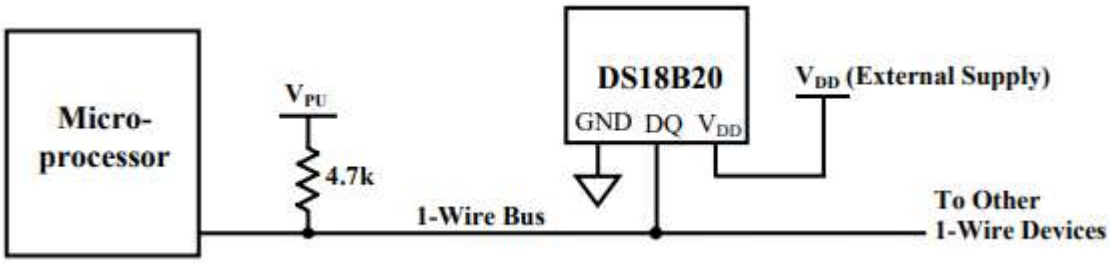


Рис. 2.6. Схема підключення датчика до плати [5]

2.6. Мова програмування С

В інформатиці С - це процедурна імперативна мова програмування: програми, написані цією мовою, складаються з математичних виразів та імперативних інструкцій, згрупованих у параметризованих процедурах, здатних керувати різними типами даних. [10]

С визначається як мова програмування високого рівня та інтегрує характеристики мов низького рівня, тобто символи, числа та адреси, які можна вказати через арифметичні та логічні оператори, що використовуються реальними машинами. С була задумана як проста та функціональна, вона також використовує численні бібліотеки для задоволення будь-яких потреб, зокрема стандартну бібліотеку С. Ці бібліотеки у вигляді файлів заголовків або файлів заголовків із суфіксом -h , може бути завантажений за допомогою директиви включення препроцесора.

Мова спочатку була розроблена Деннісом Річі в AT&T Bell Labs між 1969 та 1973 роками, з метою використовувати її для складання операційної системи UNIX, раніше зробленої Кеном Томпсоном та самим Річі під час складання PDP-7. У 1972 р. Була запущена перша система UNIX на PDC-11 DEC, повністю написаному новою мовою програмування.

Перша стандартизація С була зроблена ANSI в 1989 році (ANSI X3.159-1989), відома як С89. Ця ж версія, лише з мінімальними змінами форматування, пізніше була також стандартизована ISO у 1990 р. (ISO / IEC 9899: 1990), відома як С90. Згодом ISO випустила чотири інші версії мови С, відомі як С95 (ISO / IEC 9899 / AMD1: 1995), С99 (ISO / IEC 9899: 1999), С11 (ISO / IEC 9899:

2011 / Cor 1: 2012)) та C18 (ISO / IEC 9899: 2018). C99 вніс основні вдосконалення мови програмування, запровадивши нові типи даних, ініціалізатори, призначені для масивів, масиви змінного розміру та інші вдосконалення, запозичені з C ++.

Народжений разом з Unix, він підтримується всіма широко використовуваними операційними системами, які використовуються і в основному використовуються для створення операційних систем, мов програмування, бібліотек, ігор і для високоефективних додатків; вона відома своєю ефективністю та зарекомендував себе як орієнтир для створення системного програмного забезпечення на більшості сучасних апаратних платформ. Стандартизація мови (спочатку ANSI, а потім ISO) гарантує переносимість програм, написаних на C (стандарт, який часто називають ANSI C) на будь-якій платформі; окрім системного програмного забезпечення, вона вже давно є домінуючою мовою в ряді інших областей, що характеризуються ефективністю. Типовими прикладами є телекомунікації, управління промисловими процесами та системне програмне забезпечення в режимі реального часу. Перевага C у цих контекстах частково зменшилось після появи значних альтернатив, насамперед C ++.

C також має неабияке дидактичне значення, хоча через свою смислову складність та міцні зв’язки її семантики з функціонуванням комп'ютерних апаратних засобів вона не є особливо інтуїтивно зрозумілою мовою для новачків і, зокрема, для тих, хто не має достатнього досвіду з комп'ютерної електроніки. Якщо в середній школі та університеті колись сприймали C як основну мову через її технічне значення, то сьогодні цей вибір знаходить подальшу мотивацію у зростаючому значенні мов, що походять із C (наприклад, C ++, Java та C # , щоб дозволити студентам швидший та інтуїтивніший початковий підхід).

C - відносно мінімалістична мова програмування; її семантика використовує обмежений набір понять, які відносно прості та близькі до функціонування комп'ютерних апаратних засобів. Багато інструкцій C

безпосередньо перекладаються однією машинною мовою (наприклад, операторами автоматичного посилення та автоматичного регулювання). У мові центральну роль відіграє концепція вказівника, яка узагальнена для збігу з непрямою адресацією, способом доступу до апаратної пам'яті, характерної для всіх сучасних процесорів. Це робить С особливо ефективною мовою,вона додатково має визначену та розбірливу логічну структуру, функції у стилі Паскаля і перш за все контроль над типами (на фазі компіляції).

Синтаксис С досить універсальний і мова знаходиться у вільній формі, що дозволяє писати складні інструкції в декількох рядках коду або створювати криптовалютні та нечитабельні програми (обфузація коду). Зрештою, успіх С полягає у тому, що це мова, створена експертними програмістами, яку слід використовувати експертам-програмістам.

Ця велика свобода, синтаксична складність мови та центральна роль вказівників, які потрібно практично використовувати з перших програм, роблять її, з іншого боку, важкою мовою, яка не рекомендується для початківців, які відразу стикаються з великою кількістю проблем та помилок, які, хоч і експерту очевидні, дуже важко визначити для початківця.

Завдяки особливій ефективності коду, С використовувався для перезапису більшості коду системи UNIX, зменшуючи використання мови збирання на невелику групу функцій. Його значення, однак, зросло лише після 1978 року з публікацією Брайана Керніган та Денніса Річі книги "Програма мови С", в якій мова була визначена точно.

Подальше його широке використання призвело до народження кількох діалектів, а отже, до необхідності визначення стандарту. З цією метою влітку 1983 р. Було створено комітет, який створив стандарт ANSI (Американський національний інститут стандартів), який би визначав мову С раз і назавжди. Процес стандартизації, який зайняв шість років (набагато довше, ніж очікувалося), закінчився у грудні 1989 р., А перші екземпляри стали доступними на початку 1990-х рр. Цю версію С зазвичай називають С89. Стандарт також був прийнятий Міжнародною організацією зі стандартизації

(ISO) у 1999 році під назвою C Standard ANSI / ISO. У 1995 році була прийнята поправка 1 до стандарту C, яка, серед іншого, додала нових функцій до стандартної бібліотеки даної мови. Використовуючи C89 як основний документ із поправкою 1 та приєднавшись до використання класів Simula, Bjarne Stroustrup почав розробляти C ++.

Кінцевим результатом постійного розвитку C став стандарт, оприлюднений у 1999 році, відомий як ISO C99 (код ISO 9899).

З версією C11 (2011) деякі команди трохи переглянуті, тоді як версія C18 (2018) виправила деякі критичні аспекти C11, не ввівши, однак, жодних нових функціональних можливостей.

2.6.1. Стандартизовані можливості:

C99

- Тип даних _Bool, який дозволяє зберігати false та true булеві значення;
- Тип даних _Complex для представлення комплексних чисел;
- Тип даних _Imaginary для представлення уявних чисел;
- Довгий long тип даних int;
- Булевий тип у <stdbool.h>;
- Додаткова функціональність з плаваючою комою в <float.h>;
- Однорядкові коментарі, введені //;
- Вбудовані функції;
- Обмежувальний класифікатор, допустимий лише для обмежених покажчиків;
- Змінна довжина масиву (VLA);
- Складені літерали;
- Позначені ініціалізатори
- Дозволяється оголошувати змінні де завгодно в блоці коду, а також розміщувати інші інструкції між ними;
- Функції сім'ї vscanf;
- Правила для цілих констант;

- Цілі правила просування;
- Загальні математичні макроси в <tgmath.it>;
- Макроси зі змінною кількістю аргументів;
- Макрос va_copy;
- Специфікатор перетворення lf у функції printf ();
- Підтримка арифметики з плаваючою комою IEEE (IEC 559);
- Оператор попередньої обробки _Pragma.

C11

Стандарт c11 представив п'ять нових файлів заголовків, а саме <stdalign.h>, <stdatomic.h>, <stdnoreturn.h>, <thread.h> та <uchar.h>, а також кілька функцій, які допомогли покращити мова:

- Макроси, що стосуються специфікацій вирівнювання пам'яті з відповідним файлом заголовка <stdalign.h>, включаючи _Alignas та _Alignof, а також функцією align_alloc (Control Alignment Control);
- Додана підтримка багатопотокової передачі. Нові функції надаються бібліотекою потоків, оголошеною у файлі заголовка <thread.h>. Також додано _Atomic класифікатор у файл заголовка <stdatomic.h>;
- Анонімні структури (структури) та спілки;
- Загальні вирази за допомогою ключового слова _Generic (тип-загальні вирази);
- Покращена підтримка Unicode з типами даних char16_t (UTF-16) та char32_t (UTF-32) із відповідними функціями перетворення, оголошеними в <uchar.h>;
- Вилучена функція get (), оголошена в <stdio.h>;
- Специфікатор _Noreturn, застосовний до функцій;
- Статичні твердження за допомогою ключового слова _Static_assert (Статичні твердження);
- Функція quick_exit для завершення програми;
- Інтерфейси, що перевіряють межі;

- Особливості аналізу;
- Ексклюзивний "x" режим відкриття та створення файлів (Ексклюзивний режим створення та відкриття) ;
- Макроси для створення складних чисел у <complex.h>.

2.6.2.Функції:

- sensors.begin()
  запуск датчика температури
- MQ7.inicializar()
  ініціалізація датчика газу CO
- MQ7.update()
  скидання датчика газу та оновлення параметрів
- sensors.requestTemperatures()
  запит на отримання даних з цифрового термометру
- sensors.getTempCByIndex(0)
  отримання поточного значення температури в градусах Цельсія
- MQ7.readSensor()
  отримання даних про вміст CO в повітрі
- client.connect
  підключення до сервера
- WiFi.begin
  підключення до точки доступу Wi-fi
- WiFi.localIP()
  отримання локальної IP адреси
- WiFi.status()
  статус поточного підключення до Wi-fi

2.7. Мова програмування Java

Java - це об'єктно-орієнтована статична мова програмування на високому рівні, яка спирається на однойменну програмну платформу для виконання, спеціально розроблену для того, щоб бути максимально

незалежною від апаратної платформи виконання (шляхом компіляції в байт-коді до і потім інтерпретація за допомогою JVM) (хоча ця особливість передбачає продуктивність з точки зору обчислень, що поступається тому, що безпосередньо складені мови, такі як C і C++ або тому ідеально адаптовані до апаратної платформи). [11]

Java була створена на основі досліджень, проведених в Стенфордському університеті на початку 1990-х. У 1992 році народилася мова Дуб вироблена Sun Microsystems та створена групою експертів-розробників під керівництвом Джеймса Гослінга. Пізніше цю назву було змінено на Java з питань авторських прав: мова програмування Oak вже існувала.

Для полегшення переходу на Java для старомодних програмістів, зокрема, таких мов, як C++, основний синтаксис (структури управління, оператори тощо) майже повністю ідентичний тому, що і у C++; однак на мовному рівні функції, які вважаються непотрібними складними, що сприяють введенню певних помилок під час програмування, таких як арифметика вказівника та успадкування декількох класів, не були введені.

Спочатку Sun вирішила використовувати цей новий продукт для створення складних додатків для невеликих електронних пристроїв; Лише в 1993 році з вибухом Інтернету Java почала використовуватись як інструмент для початку програмування в Інтернеті.

У той же час корпорація Netscape оголосила про вибір обладнання для віртуальної машини Java (JVM). Це знаменує революцію у світі Інтернету: завдяки аплетам веб-сторінки стали інтерактивними на рівні клієнтів, тобто програми запускаються безпосередньо на машині користувача Інтернету, а не на віддаленому сервері. Наприклад, користувачі могли використовувати ігри безпосередньо на веб-сторінках і скористатися динамічними та інтерактивними чатами.

Java була офіційно оголошена 23 травня 1995 року на SunWorld.

13 листопада 2006 року Sun Microsystems випустила реалізацію компілятора Java та віртуальної машини за ліцензією GPL. Не всі платформи Java безкоштовні. Вільне середовище Java називається IcedTea.

8 травня 2007 року Sun також опублікував бібліотеки, за винятком деяких компонентів, які не належать йому, під ліцензією GPL, зробивши Java мовою програмування, реалізація посилань якої безкоштовна.

Мова визначається документом, який називається Специфікація мови Java, часто скорочується JLS. Перша редакція документа була опублікована в 1996 році. З тих пір мова зазнала численних модифікацій та доповнень, доданих час від часу в наступних виданнях. Наприкінці 2018 року найновішою версією специфікацій є Java SE 11 Edition.

Один з основних принципів мови виражається девізом WORA (написати один раз, запустити скільки завгодно): складений код, який виконується на одній платформі, не потрібно перекомпілювати для виконання на іншій платформі. Фактично продукт компіляції знаходиться у форматі, який називається байт-кодом, який може бути виконаний будь-якою реалізацією віртуального процесора під назвою Java Virtual Machine.

Станом на 2014 рік, Java є однією з найбільш використовуваних мов програмування у світі, особливо для клієнтсько-серверних додатків. За оцінками, кількість розробників становить близько 9 мільйонів.

2.7.1. Принципи

Java створена для виконання п'яти основних завдань:

- "простою, об'єктно-орієнтованою та знайомою";
- "надійною і безпечною";
- незалежною від платформи;
- містити інструменти та бібліотеки для мережі;
- бути розробленою для безпечного запуску коду з віддалених джерел.

Програми, написані мовою Java, після початкової фази компіляції з отриманням так званого байтового коду, призначені для виконання на

платформі Java через фазу інтерпретації (з цієї причини мову Java також називають напівінтерпретованою) Віртуальна машина та Java яка працює на ній мають доступ до стандартних API бібліотеки. Ці два кроки забезпечують рівень абстракції, що дозволяє додаткам бути повністю незалежними від апаратної системи, на якій вони будуть запускатися.

Реалізація платформи java - це середовище виконання JRE, необхідне для виконання скомпільованої програми, тоді як для розробки програм на Java, починаючи з вихідного коду, потрібен комплект Java Development Kit (JDK), який також включає JRE.

Сама мова визначає лише мінімальну частину бібліотек, яку можна використовувати в поєднанні з самою мовою. Залишок визначається платформою, на якій запускається програма.

Oracle пропонує три офіційні платформи, кожна призначена для різних областей:

- Платформа Java, стандартне видання;
- Платформа Java, Micro Edition;
- Java EE.

2.7.2. Переносимість

Виконання програм, написаних на Java, повинно мати подібну поведінку в різних контекстах виконання. Наприклад, вона пропонує уніфікований синтаксис для визначення критичних розділів, завдання, яке в інших мовах зазвичай відбувається за допомогою сторонніх бібліотек або системних примітивів. Крім того, це практично не залишає місця для невизначеної поведінки або для реалізації, що залежить від впровадження.

Мовні специфікації вимагають середовища виконання, яке контролює виконання програми і яке забороняє певні операції, які в іншому випадку були б незахищеними. Вони явно посилаються на віртуальну машину Java, вказуючи її як типового одержувача байт-коду, отриманого при первинній компіляції програми Java, а насправді компілятор javac, що входить до JDK,

компілюється саме в байт-код. Однак можна компілювати в різні архітектури, адже можна створити конкретний об'єктний код певної операційної системи за допомогою спеціального компілятора, наприклад, колекції компілятора GNU. В принципі, ви повинні мати можливість написати програму один раз і запустити її де завгодно, звідси відомий слоган Sun: «пишіть один раз, запускайте безліч». Переносимість - технічно складна мета, яку можна досягти, а успіх Java в цій галузі викликає суперечки. Хоча насправді можна писати в програмах Java, які поводяться послідовно на багатьох різних апаратних платформах, вони залежать від віртуальних машин, які є програмами самі по собі і які неминуче мають свої помилки, відмінні один від одного.

2.7.3. Бібіліотеки

Крім того, програміст може використовувати довільну кількість сторонніх бібліотек. Ці бібліотеки, що містяться в різних пакетах, використовуються програмістом для використання певних загальних методів або атрибутів для спрощення та стандартизації програм та підвищення їх читабельності для програмістів. Існує безліч пакетів, які програмісти можуть використовувати на мові Java. Наприклад, є пакети:

- ввід і вивід (java.io);
- для математичних методів і констант (java.Math);
- для створення аплетів (java.applet);
- створювати мережеві програми (javax.net);
- для виводу (javax.print);
- для безпеки (java.security та javax.security).

2.7.4. Об’єктивність

На відміну від C ++, Java є більш об'єктно-орієнтованим. Всі дані і дії згруповані в класи об'єктів. Винятком з повної об'єктивності є примітивні типи. Це було свідоме рішення мовних дизайнерів збільшити швидкість. Через це Java не є повністю об'єктно-орієнтованою мовою.

У Java всі об'єкти є похідними від основного об'єкта який має просту назву - Object, який являє собою основний шаблон з властивостями та поведінкою.

Хоча множинне спадкування є вперше в C ++, в Java можливо тільки одне спадкування, що виключає можливість конфліктів між членами класу (методами і змінними), які успадковуються від базових класів.

2.7.5. Безпека

Розробники мали намір замінити Java на C ++, спадкоємицю об'єкта C. Розробники почали з аналізу властивостей C ++, які призводять до більшості помилок, для створення простого, безпечного і безвідмовного мови програмування.

В Java існує система виключень або ситуацій, коли програма стикається з несподіваними труднощами, такими як:

- операції над елементом масиву які знаходяться за його межами або операції на порожнім елементом;
- читання з каталогу до якого немає доступу або з невірною URL адресою;
- введення невірних даних користувачем.

Однією з особливостей концепції віртуальної машини є те, що помилки не приводять до повної відмови системи. Крім того, є інструменти, які підключаються до середовища виконання і при виникненні виключень, коли виникає певний виняток, записують інформацію з пам'яті для налагодження програми. Ці автоматизовані інструменти обробки винятків надають базову інформацію про винятки в додатках Java.

Проте, мова програмування Java не рекомендується для використання в системах, відмова яких може призвести до смерті, травми або серйозним фізичним травм через можливу відмову за рахунок певної ненадійності програм, написаних на мові Java.

2.7.6. Автоматичне керування пам’яттю

У Java є у використані автоматичний збирач сміття GC, який дозволяє управляти пам’яттю протягом життєвого циклу об'єкта. Програміст визначає коли створювати об’єкт при написанні коду програми, а віртуальна машина відповідає за звільнення пам'яті коли створені об’єкти перестають використовуватися. Коли немає посилань на певний об'єкт, збирач сміття може автоматично видалити його з пам'яті. Проте, витоку пам'яті все ще можуть виникати, якщо код, написаний програмістом, має посилання на об'єкти, які більше не потрібні, наприклад, об'єкти, що зберігаються в існуючих контейнерах.

Збір сміття дозволений в будь-який час. В ідеалі це відбувається під час бездіяльності програми. Збір сміття автоматично ініціюється, коли в купі не вистачає вільної пам'яті для розміщення нового об'єкта, що може привести до зависання на кілька секунд. Отже, існують реалізації віртуальної машини Java з збирачем сміття, спеціально розроблені для програмування систем реального часу.

Java не підтримує покажчики стилю C / C ++. Це зроблено для забезпечення безпеки і надійності, щоб збирач сміття міг переміщати об'єкти покажчика.

2.7.7. Методи:

- send

  відправка повідомлення з сервера
- downService

  зупинити сервіс
- getString

  отримати строку
- parseInt

  парсити на ціле число
- getConnection

Підключитись до БД

- accept

  приняти

- serverList.add

  додати нову нитку підключення

- getInputStream

  отримати вхідний потік

- getOutputStream

  отримати вихідний потік

2.8. База даних MySQL

MySQL або Oracle MySQL - це система управління реляційними базами даних (RDBMS), що складається з клієнта командного рядка та сервера, доступних як для Unix, Unix-подібних систем та для Windows; основними референтними платформами є Linux та Oracle Solaris. [16]

Вільне програмне забезпечення, випущене з подвійною ліцензією, включаючи загальну публічну ліцензію GNU, розроблене таким чином, щоб максимально відповідати стандартам ANSI SQL і ODBC SQL, а системи та мови програмування, які підтримують його, дуже багато: ODBC, Java, Mono, .NET, PHP, Python та багато інших.

Платформи LAMP і WAMP містять MySQL для впровадження серверів для управління динамічними веб-сайтами; також багато успішних систем управління вмістом, таких як WordPress, Joomla, Drupal і TikiWiki, народжуються з підтримкою MySQL за замовчуванням. Програмне забезпечення MediaWiki, яке керує сайтами проекту Wikimedia, базується на базі даних MySQL.

2.8.1. Можливості сервера MySQL:

- простий у використані та встановленні;
- здатний підтримувати велику кількість користувачів які можуть одночасно працювати з БД;

- велика кількість рядків у таблицях, до 50 млн;
- висока швидкість виконання команд;
- ефективна і проста система безпеки.

2.8.2. Запити:

- INSERT INTO

  додати нові записи в таблицю
- SELECT DISTINCT

  вибір унікальних записів
- SELECT

  вибір записів з таблиці
- UPDATE

  зміна даних в таблиці

2.9. Google Map API

За останні кілька років API серед веб-проектів отримали досить великий попит, особливо в галузі карт. Сьогодні складно уявити роботу веб-розробників без такого інструменту як API, які також стали дуже ефективним елементом в маркетингу багатьох шарів бізнесу. У зв'язку з цим потрібно розібратися з тим, що таке API, їх можливості і переваги, а також основний перелік прихованих функцій. [17]

API Google Карти - це програмований, картографічний сервіс, який надається компанією Google. Його можна легко розмістити на своєму сайті і періодично відзначати своє місце положення, щоб клієнти могли легко визначити ваше місце розташування і без зайвого клопоту відшукати офіс.

Сам сервіс представлений у вигляді набору протоколів, за рахунок яких програмісти і веб-розробники можуть як «по цеглинці» збирати різні додатки. Ще зовсім недавно API був тісно пов'язаний з ОС і додатками на робочому столі. Але за останні кілька років тренд настільки виріс, що API став незамінним інструментом в сфері веб.

Перш ніж переходити до професійного використання API Google Карти потрібно мати певний рівень знань в об'єктно-орієнтованому програмуванні. З цією метою відмінно підійде Java. Для повноцінної роботи потрібно отримати ключ (для додатків, які працюють не на сервері, цей пункт не є обов'язковим). За допомогою унікального ключа компанія Google може не тільки відстежувати додатки працюють з сервісом API, але і в разі потреби зв'язатися з власником програми. Дуже зручно і практично для роботи з картами використовувати накладення (шари), за рахунок яких за координатами можна будувати геометричні фігури - примітиви, з їх допомогою відбувається візуалізація на карті.

Аналогом точки вважається marker. Щоб позначити положення маркера на карті досить вказати два сферичних координата. Лінія (line) задається координатами її початку і кінця, а ось полігон (polygon) за допомогою безлічі сферичних координат з положення вузлових точок. Таким чином, важливі точки на карті позначаються маркером, розрахувати відстань між двома ключовими точками можна за рахунок лінії, а відобразити площа, яку ми обчислюємо можна за допомогою полігону.

На сервісі передбачені додаткові фігури, такі як коло (circle) - задається радіусом і координатами центру, прямокутник (rectangle) - задається сферичними координатами вершин і т. Д. Для спрощення розрахунків відстані між двома точками або ж обчислення площі полігону потрібно використовувати спеціальну бібліотеку - Geometry.

Найчастіше у початківців розробників виникають складнощі зі сферичними координатами. Адже щоб правильно вирішити геометричні завдання на площині потрібно зробити спеціальний перехід. З цією метою знадобляться вбудовані функції сервісу. Досить вражаюче завжди виглядають анімації на картах. Наприклад: можна змоделювати рух літака або автомобіля за вказаною траєкторії.

2.9.1. Основні переваги сервісу:

• Даний сервіс досить простий та зрозумілий для використання у власних проектах. Користувачеві потрібно лише надавати дані про місце знаходження пристрою, а сервіс розмістить всі необхідні дані в маркері який буде встановлено за відповідними координатами;

• Завдяки широкому використанню сервісу в повсякденному житті інтерфейс дуже зрозумілий та дозволить користувачам легко знайти потрібну інформацію на карті;

• Функціональність сервісу може бути змінена під конкретні вимоги користувачів;

• Можливість залишати відгуки які будуть відображені під контактною інформацією компанії;

• Створення власних маршрутів з великою кількістю проміжних зупинок;

• Синхронізація карт між різними пристроями;

• Збереження на карті місць з частим відвідуванням;

• Можливість дізнатися точну поштову адресу, індекс, контактний номер і т. Д .;

• Режим роботи карт;

• Можливість вносити поправки в карти Google.

У підсумку, API Google Maps вважається досить потужною платформою, яка з кожним днем удосконалюється і набирає обертів популярності. Правильне використання всіх функцій сервісу дозволяє зробити будь-яку подорож.

2.9.2. Методи API Google Maps:

- getSupportFragmentManager
  об’єкт карти
- mMap.moveCamera
  показати розташування за координатами

- postMarker

  показати маркер
- MarkerOptions

  додати параметри маркеру
- addMarker

  додати маркер на карту
- MarkerOptions().position

  додати розташування в параметри маркеру
- Remove

  видалити маркер з карти

2.10. Android

Android - це операційна система для мобільних пристроїв, розроблена компанією Google LLC і заснована на ядрі Linux, яка вважається належним чином вбудованим дистрибутивом Linux, а не unix-подібною системою або дистрибутивом GNU / Linux (оскільки майже всі утиліти GNU замінені на програмне забезпечення на Java), призначене в основному для вбудованих систем, таких як смартфони та планшети, зі спеціалізованим інтерфейсом користувача для телевізорів (Android TV), автомобілів (Android Auto), наручних годин (Wear OS), окулярів (Google Glass) та інших. [11][13]

Користувальницький інтерфейс Android базується на концепції прямого маніпулювання, згідно з яким моно- та багатоконтактні входи, такі як прокручування, натискання та натискання на екран використовуються для маніпулювання об'єктами, видимими на екрані. розроблений так, щоб бути швидким та зі плавним інтерфейсом. Внутрішні апаратні датчики, такі як акселерометри, гіроскопи та датчики наближення, використовуються деякими програмами для реагування на дії користувача, наприклад, налаштування екрана від вертикалі до горизонталі залежно від орієнтації пристрою або дозволу користувача керувати транспортним засобом у віртуальній гонці, повертаючи пристрій, імітуючи управління кермом.

Android (на відміну від iOS) дозволяє очистити кеш-пам'ять як в цілому, так і індивідуально для конкретного додатка / послуги. Існує також команда, яка дозволяє звільнити оперативну пам’ять від запуску, але не є істотними програмами / послугами. Окрім певного кешу, команда також дозволяє видалити дані, що зберігаються додатком: на практиці ви отримуєте додаток так, ніби ви щойно встановили його. Нарешті, меню, яке можна використовувати для припинення програми / послуги, а не для видалення, маючи можливість вибору системних служб (розширений вигляд).

Головний екран, позначений піктограмою, що представляє будинок, схожий на робочий стіл Windows, як бачать користувачі після запуску, або натиснувши кнопку «Головна». Тут розміщені посилання на запуск програм у вигляді іконок та віджети з різними функціями; є віджети, які показують різні стилі годиника, показують останні відео YouTube, відображають інформацію про погоду. Фактичний головний екран може бути інтегрований з інших сторінок, включаючи користувач може прокручувати вперед і назад.

Класичний компонент світу Android - це Launcher (літ. "Запуск") або системний додаток, який контролює та керує головним екраном, по-друге, ярлики, панель додатків, панель внизу та рядок стану, меню сповіщень та швидкі налаштування. Окрім типових, є численні сторонні додатки, які пропонують широкий спектр налаштувань.

Меню вимкнення - це область елементів керування, які з’являються при натисканні кнопки ввімкнення / вимкнення (команди зазвичай: вимкнення, перезапуск, режим літака / офлайн, аварійний режим), іноді внизу супроводжуються звуковими командами. Якщо натиснути іншу клавішу або натиснути будь-яку іншу область, меню зникає.

Завжди у верхній частині екрана знаходиться смужка стану, на якій відображається інформація про пристрій та його підключення. Натомість у нижній частині екрана зазвичай розташована нижня смужка стандартних програм: це залишається фіксованим під час прокрутки екранів. Перетягуючи рядок стану вниз, з'являється екран сповіщень, на якому програми можуть

відображати сповіщення про важливу інформацію або оновлення, такі як електронна пошта або текстове повідомлення, щоб не одразу перервати користувача. Ранні в Android ці сповіщення можна було використовувати лише для відкриття програми, але останні оновлення забезпечують більшу функціональність, таку як можливість зателефонувати на номер безпосередньо з повідомлення про пропущений дзвінок, не відкриваючи програму Повідомлення по телефону зберігаються, поки користувач не прочитає або видалить їх.

Платформа використовує базу даних SQLite, спеціалізовану бібліотеку SGL для двовимірної графіки (замість класичного X-сервера інших дистрибутивів Linux) та підтримує стандарт OpenGL ES 2.0 для тривимірної графіки. Програми запускаються через віртуальну машину Dalvik - віртуальну машину, пристосовану для використання на мобільних пристроях.

Android був розроблений в основному для смартфонів і планшетів, але відкритий і настроюваний характер операційної системи дозволяє використовувати його на інших електронних пристроях, включаючи ноутбуки та нетбуки, смартбуки, електроні книг, камери та смарт-телевізори (Google TV). Ринок "розумних речей" останнім часом значно зріс до того, що стимулює творчість людей. Прикладом є смарт-годинник, оснащений «легкою» операційною системою Android, навушниками, автомобільними програвачами компакт-дисків та DVD, смарт-окулярами (Google Glass), холодильниками, супутниковими навігаційними системами для автомобілів, системами домашньої автоматизації, ігровими консолями, дзеркала, камери, MP3 / MP4 програвачі та бігові доріжки.

2.10.1. Програми

Програми (або додатки) - це найбільш загальна форма програмного забезпечення, яке можна встановити на Android. Їх можна завантажити як з офіційного каталогу Google Play, так і з інших каталогів, таких як Amazon Appstore Amazon.com або F-Droid, який містить лише безкоштовне ліцензійне програмне забезпечення. Програми для Android можна встановити

безпосередньо з файлу APK, наданого дистриб'ютором програмного забезпечення.

З міркувань безпеки IT-додатків надається система сертифікації, яка перевіряє цілісність встановленого пакету та авторство стосовно дистриб'ютора програмного забезпечення, акредитованого Google.

Android оснащений низкою попередньо встановлених додатків, починаючи від браузера до аналогового FM-радіо, від календаря до програми Gmail, від калькулятора до супутникового навігатора повороту, а також включає голосовий пошук Google з можливістю вибору мови.

2.10.2. Управління пам’яттю

Оскільки пристрої Android, як правило, працюють від акумулятора, Android призначений для управління процесами з метою мінімізації споживання енергії. Коли програма не використовується, система обмежує використання ресурсів, так що, хоча вона доступна для негайного використання, вона не використовує значну кількість ресурсів акумулятора чи процесора. Android автоматично керує програмами, які зберігаються в пам'яті: коли пам'яті недостатньо, система починає, невидимо, автоматично закривати неактивні процеси, починаючи з тих, які були неактивними довше.

2.10.2. Режим розробника

Android надає довгий список вдосконалених команд, корисних для включення або відключення або налаштування чи моніторингу поведінки, функцій, служб, налаштувань, інформації. Це меню вимкнено за замовчуванням, але його можна активувати, швидко натискаючи 7 разів на "Номер збірки" у Налаштуваннях \ Інформація про телефон \ Інформація про програмне забезпечення. Згодом його можна деактивувати, використовуючи перемикач увімкнення / вимкнення поруч із новим елементом, який є з'являються в налаштуваннях (Параметри розробника).

Наприклад, в меню є команда, яка дозволяє вам підключити пристрій до ПК за допомогою кабелю USB.

2.10.4. Технічні особливості

Основною апаратною платформою Android є ARM (архітектури ARMv7 та ARMv8-A), архітектури x86 та x86-64 офіційно підтримуються також у версіях, що відповідають Lollipop. Неофіційний проект Android-x86 забезпечив підтримку архітектур x86 до офіційної підтримки. Архітектури ARMv5TE та MIPS32 / 64 також історично підтримувалися, але згодом були видалені. Android-пристрої з процесорами Intel почали з'являтися в 2012 році, включаючи телефони та планшети. Отримуючи підтримку 64-бітових платформ, Android спочатку був створений для роботи в 64-бітних архітектурах x86, а потім ARM64. 64-розрядні варіанти всіх платформ від Android 5.0 підтримуються на додаток до 32-розрядних варіантів усіх платформ.

Мінімальні вимоги оперативної пам'яті для пристроїв з ОС Android 7.1 варіюються від 2 ГБ для найкращого обладнання, до 1 ГБ для пристроїв середнього класу, до абсолютного мінімуму 512 МБ для 32-бітних смартфонів з нижчими характеристиками. Рекомендація для Android 4.4 повинна мати щонайменше 512 Мб оперативної пам'яті, тоді як для пристроїв з «низькою оперативною пам'яттю» 340 Мб є мінімальна необхідна кількість, яка не включає пам'ять, виділену для різних апаратних компонентів, таких як процесор базової смуги. Для Android 4.4 потрібен 32-розрядний процесор архітектури ARMv7, MIPS або x86 (останні два - через неофіційне перенесення), а також графічний процесор, сумісний з OpenGL ES 2.0 (GPU). Android підтримує OpenGL ES 1.1, 2.0, 3.0, 3.1 та найновішої версії, 3.2 та Android 7.0 Vulkan (а для деяких пристроїв доступна версія 1.1). Деякі додатки можуть явно вимагати певної версії OpenGL ES і для запуску цих програм потрібне відповідне обладнання GPU.

Пристрої Android містять безліч додаткових апаратних компонентів, включаючи камери чи відеокамери, GPS, датчики орієнтації, спеціальні контрольні ігри, акселерометри, гіроскопи, барометри, магнітометри, датчики наближення, датчики тиску, термометри та сенсорні екрани. Деякі апаратні

компоненти не потрібні, але вони стали стандартними в деяких класах пристроїв, таких як смартфони, і якщо є додаткові вимоги, вони застосовуються. Спочатку було потрібно деяке інше обладнання, але ці вимоги були повністю знижені або усунені взагалі. Наприклад, оскільки Android спочатку був розроблений як операційна система телефону, необхідне обладнання, таке як мікрофон, з часом перестав вважатись обов'язковою складовою. Android також потребує камери автофокусування, яка може бути замінена на камери з фіксованим фокусуванням, якщо така є, оскільки цю камеру як вимогу повністю виключили, коли Android почали використовувати на телеприставках.

Окрім роботи на смартфонах та планшетах, кілька постачальників запускають Android на звичайному пристрої на звичайному ПК із клавіатурою та мишкою. Окрім їх доступності на комерційному обладнанні, подібні версії Android для ПК безкоштовно доступні в рамках проекту Android-x86, включаючи користувацький Android 4.4.

Висновки до розділу 2

В даному розділі було розглянуто технології та технічні пристрої які будуть використані при роботі над системою аналізу загазованості повітря для Smart city. Було проведено огляд використаних методів в мові С для Arduino, в мові Java для програмування серверної частини яка буде виконувати запити в базу даних, та методи для роботи з Android додатком.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1. Визначення вимог і завдань

Необхідно створити систему програм, яка буде яка буде здійснювати аналіз, передачу, зберігання та наглядно демонструвати результати роботи.

Для цього потрібно створити 3 компоненти системи:

1. Пристрій аналізу повітря з можливістю передачі даних.
2. Серверна частина для обробки та зберігання даних.
3. Клієнтський додаток на базі Android для відображення даних.

Програми мають містити наступний функціонал:

1. Аналіз стану повітря та температури.
2. Передача даних з пристрою моніторингу на сервер.
3. Отримання та зберігання даних.
4. Запит даних з мобільного додатку.
5. Відображення даних на карті за допомогою маркерів.

3.2. Вибір мови програмування

Для розробки програмного продукту було обрано такі мови програмування:

1. Мова Java – для серверної та клієнтської частини за рахунок того, що мова об’єктно-орієнтовна, систему можна зробити гнучкою для додання нових функцій, швидкості роботи та портативності, великої кількості бібліотек.
2. Мова C – для пристрою моніторингу за рахунок компактності та зручності у використані з вибраними датчиками.

3.3. Вибір допоміжних компонентів програми

- Для відображення даних на карті використовуються карти від Google. Взаємодія з ними відбувається через Google Map API.

- Для роботи з датчиками використовуються бібліотеки ESP8266WiFi, ESP8266WebServer, OneWire, DallasTemperature, та оптимізована бібліотека MyMQ7 на базі бібліотеки MQUnifiedsensor
- Для обміну інформацією між компонентами системи використовуються socket.

3.4. Опис алгоритму

Розглянемо алгоритм реалізований в системі. Він складається з таких кроків:

- Запуск сервера;
- Запуск пристрою аналізу;
- Встановлення socket з'єднання;
- Передача даних від пристрою аналізу до сервера;
- Запуск мобільного додатку;
- Запит на сервер про наявні точки моніторингу та останні дані;
- Обробка запиту, формування відповіді та відправка даних на клієнт;
- Прийом даних клієнтським додатком;
- Обробка даних та запит у Google Map API;
- Відображення даних на карті за допомогою Google Map API.

3.5. Вимоги до технічного забезпечення

1) Основні вимоги до устаткування:
    - Ноутбук або ПК з 4 ядерним процесором, об'єм оперативної пам'яті - 4 Гб, об'єм вільного дискового простору – 2 Гб, постійне підключення до мережі інтернет;
    - Мобільний телефон з GPS на базі Android v7 та вище;
    - NodeMcu v3, DS18B20, MQ-7, макетна плата, конектори та резистор на 4700 Ом.
2) Вимоги до програмного забезпечення:
    - операційна система Windows 10 або Mint;
    - Java 8;

- MySQL – база даних;
- Встановлений мобільний додаток для відображення даних;
- Arduino IDE, Android Studio, Intellij Idea.

3.6. Інструкція користувача по первинному налаштуванні

3.6.1. Налаштування пристрою моніторингу

Для початку роботи потрібно мати зібраний пристрій моніторингу як на рис. 3.1. схема Додаток 2.

Для датчика DS18B20 підключити пін GND до GND NodeMcu, пін даних через резистор до D2 на платі, та VCC до 3.3 в на платі через іншу ніжку резистора.

Для датчика MQ – 7 підключити A0 до A0 плати, VCC до 3.3 в, GND до GND NodeMcu.

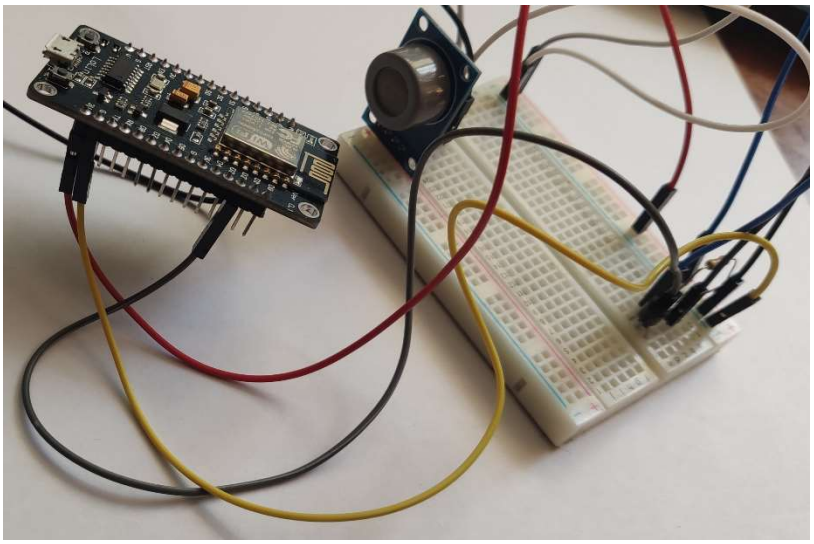


Рис. 3.1. Пристрій моніторингу в зібраному вигляді

Для коректної роботи потрібно замінити налаштування підключення до Wi-Fi – в методі connectTOWiFi. Це потрібно для підключення та передачі даних (рис. 3.2). Також при зміні порту сервера потрібно замінити порт сервер в коді пристрою.

```
void connectTOWiFi(){
  delay(100);
  WiFi.begin("sour_cream", "2yoghurts");

  while (WiFi.status() != WL_CONNECTED) {
    delay(CONNECT_TO_WIFI_RETRY_DELLAY);
    Serial.println("Waiting to connect…");
  }
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Рис. 3.2. Програмування технічного засобу

3.6.2. Налаштування серверної частини

Серверна частина потребує налаштування бази даних, ввід логіна та пароля для підключення з додаванням потрібних таблиць при відсутності. При необхідності змінити порт підключення.

3.6.3. Налаштування мобільного додатку

Для додатку потрібно налаштувати порт та адресу серверу який зберігає та обробляє дані(рис. 3.3). Для цього потрібно у класі MapsActivity замінити адресу і порт.

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
    public static String ipAddr = "192.168.1.170";
    public static int port = 7047;

    public double l;
    public double w;
```

Рис. 3.3. Програмування мобільного додатку

Висновки до розділу 3.

В даному розділі розглянуто технічні вимоги до кінцевого продукту та опис алгоритму з інструкцією по налаштування для впровадження системи.

Так за допомогою інструкції по налаштуванню можна запровадити дану систему для використання на підприємствах або у інших установах.

РОЗДІЛ 4
ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ

4.1. Робота пристрою аналізу

Система для аналізу загазованості повітря була успішно розроблена та протестована.

На рис. 4.1. зображено спроби підключення до мережі та вивід адреси пристрою в даній мережи при успішному підключенні.

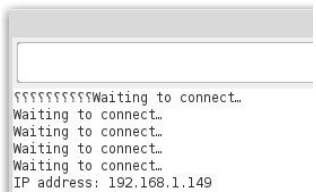


Рис. 4.1. Підключення до мережі Wi-Fi

На рис. 4.2. результати виміру та спроби передати даних, при відсутності підключення до сервера ми отримуємо повідомлення про помилку підключення. Перша цифра це унікальний номер пристрою який генерується на основі MAC адреси, далі йде поточне значення вмісту CO в одиницях ppm та температура повітря у місці виміру в градусах Цельсія.

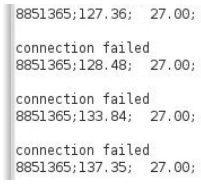


Рис. 4.2. Вивід даних та помилки

4.2. Робота серверної частини системи

Зі сторони сервера відбувається прийом даних та їх обробка згідно з завданням, а саме передача мобільним додаткам та зберігання. На рис. 4.3. відображена сортована таблиця отриманих даних, де перший стовбець це унікальний номер пристрою, другий вміст CO, третій температура, четвертий

та п’ятий відповідають за координати пристрою аналізу, та остання це дата отримання результатів виміру.

	unique_id	CO	temp_data	lenth	width	data_time
1	8851365	193.24	23.69	50.448	30.4515	2020-05-15 18:00:11
2	8851365	193.01	23.69	50.448	30.4515	2020-05-15 18:00:09
3	8851365	193.01	23.75	50.448	30.4515	2020-05-15 18:00:08
4	8851365	193.01	23.69	50.448	30.4515	2020-05-15 18:00:07
5	8851365	193.01	23.75	50.448	30.4515	2020-05-15 18:00:06
6	8851365	193.01	23.69	50.448	30.4515	2020-05-15 18:00:05
7	8851365	193.01	23.69	50.448	30.4515	2020-05-15 18:00:04
8	8851365	193.01	23.69	50.448	30.4515	2020-05-15 18:00:03
9	8851365	193.01	23.69	50.448	30.4515	2020-05-15 18:00:02
10	8851365	193.01	23.69	50.448	30.4515	2020-05-15 18:00:01
11	8851365	192.78	23.62	50.448	30.4515	2020-05-15 18:00:00
12	8851365	192.32	23.69	50.448	30.4515	2020-05-15 17:59:58
13	8851365	192.09	23.69	50.448	30.4515	2020-05-15 17:59:57
14	8851365	191.86	23.69	50.448	30.4515	2020-05-15 17:59:56
15	8851365	192.09	23.69	50.448	30.4515	2020-05-15 17:59:55

Рис. 4.3. Таблиця даних на стороні сервера

4.3. Робота мобільного додатку

Мобільний додаток використовується як для відображення результатів на карті так і для налаштування пристрою виміру.

Налаштування пристрою аналізу проходить у такому порядку:

1. Отримання унікального номеру пристрою.
2. Підключення та передача номеру пристрою, даних аналізу на сервер.
3. Ввід унікального номеру пристрою у вікно налаштувань яке викликається кнопкою GO в мобільному додатку.
4. Натискання кнопки SET для відправки номеру пристрою та поточних значень координат мобільного пристрою для закріплення пристрою за розташуванням.

Рис. 4.4. показує вікно налаштувань координат пристрою аналізу. Кнопка UP дозволяє оновити дані на карті.

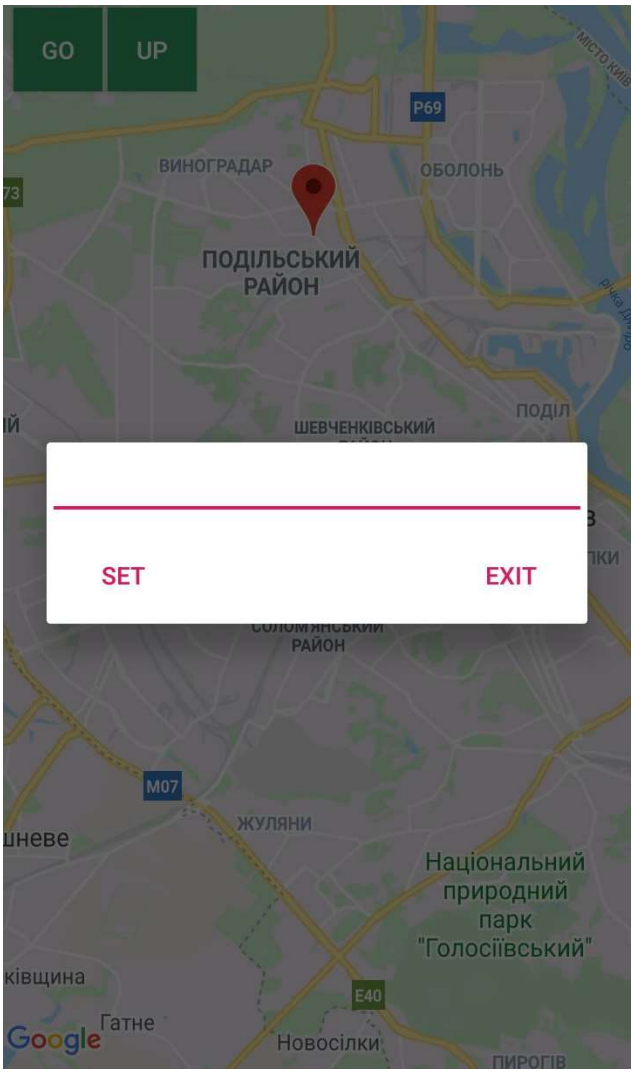


Рис. 4.4. Меню налаштування позиції пристрою аналізу

Даний додаток дозволяє як повертати карту у потрібному напрямку, знаходити точне розміщення пристроїв аналізу.

Для тесту було введено в базу даних декілька пристроїв, але за наявності тільки одного фізичного пристрою дані інших не будуть змінюватись при оновлені. Рис. 4.5. та рис. 4.6. показують вивід даних у текстовому форматі, як опис маркера з номером пристрою та останніми результатами аналізу.

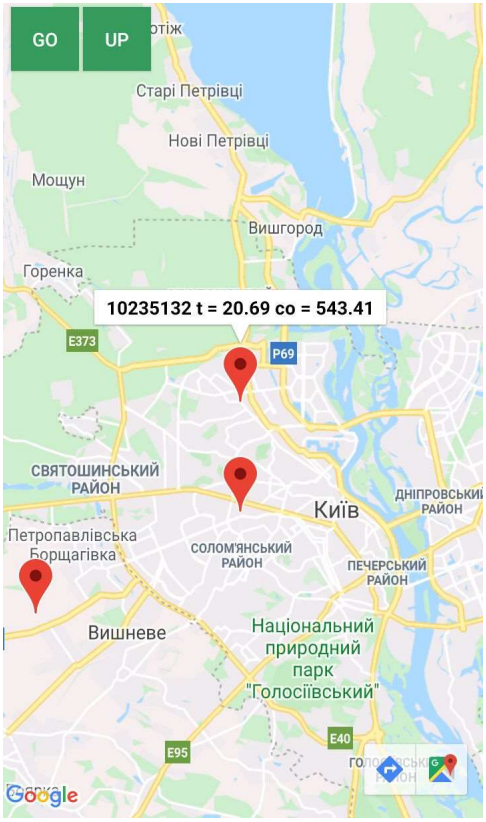


Рис. 4.5. Маркер з сталими значеннями пристрою без оновлень

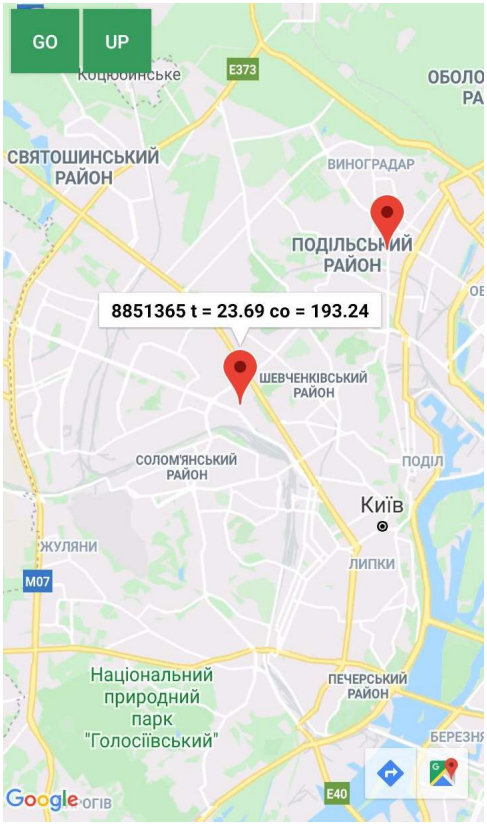


Рис. 4.6. Фізично підключений пристрій

Висновки до розділу 4.

В даному розділі продемонстровані результати роботи системи аналізу загазованості повітря.

Так представлено результати виміру на пристрої аналізу, запис даних в базі даних та демонстрація даних на мобільному пристрої, що показує працездатність системи.

ВИСНОВКИ

У рамках дипломного проекту були досягнуті перераховані нижче результати:

1. Був проведений аналіз існуючих рішень для даної проблеми з забруднення повітря.
2. Було розглянуто засоби для розробки програмного забезпечення для системи так і технічна сторона даного питання така як вибір устаткування.
3. Створена система аналізу повітря з відображенням результатів аналізу у мобільному додатку. В усіх випадках програма показала свою працездатність.

Враховуючи все вище зазначене дана система може бути впроваджена як на території установи, закладу освіта для контролю за станом повітря так і використана як навчальний матеріал у курсі по програмуванню засобів інтернету речей.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Kunak [Електронний ресурс] - https://www.kunak.es/en/
2. Aeroqual [Електронний ресурс] - https://www.aeroqual.com/
3. IQair [Електронний ресурс] - https://www.iqair.com/
4. Ромашко С. М. Конспект лекцій з дисципліни «Комп'ютерні мережі і телекомунікації» - Львів: ЛРІДУ НАДУ, 2006. - 61с.
5. DS18B20 [Електронний ресурс] - https://html.alldatasheet.com/html-pdf/227472/DALLAS/DS18B20/216/1/DS18B20.html
6. MQ-7 [Електронний ресурс] - https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MQ-7%20Ver1.3%20-%20Manual.pdf
7. Rossitza I. Chichkova, Leon D. Prockop. Carbon monoxide intoxication: An updated review // Journal of the Neurological Sciences. — 2007. — Vol. 262, iss. 1—2. — P. 122–130
8. S. Brandon, J. Sydney Smith. Acute carbon monoxide poisoning—3 years experience in a defined population // Postgraduate Medical Journal. — 1970. — Vol. 46, iss. 532. — P. 65–70.
9. Окись углерода (угарный газ) // Вредные вещества в промышленности. Справочник для химиков, инженеров и врачей / Под ред. Н. В. Лазарева и И. Д. Гадаскиной. — 7-е изд. — Л.: Химия, 1977. — Т. 3. — С. 240-253. — 608 с.
10. Brian W. Kernighan and Dennis M. Ritchie: The C Programming Language, 2nd ed., Prentice Hall, 1988, с. 3.
11. Б. Эккель "Философия Java" Сп.Б.: "Питер" 2009 – 638с.
12. NodeMcu [Електронний ресурс] - https://www.handsontec.com/pdf_learn/esp8266-V10.pdf
13. Филлипс Б.Стюарт К.Марсикано К. "Android. Программирование для профессионалов. 3-е издание": "Питер Пресс" 2017 – 688с.

14. Белов А.В. "Программирование ARDUINO. Создаем практические устройства".: Наука и Техника, 2018. -272 с.
15. Програма Cisco Networking Academy, CCNA 1 і 2 Companion Guide переглянуто третє видання, Р.480.
16. MySQL [Електронний ресурс] - https://www.mysql.com/
17. Google map API [Електронний ресурс] - https://developers.google.com/maps

ДОДАТОК 1

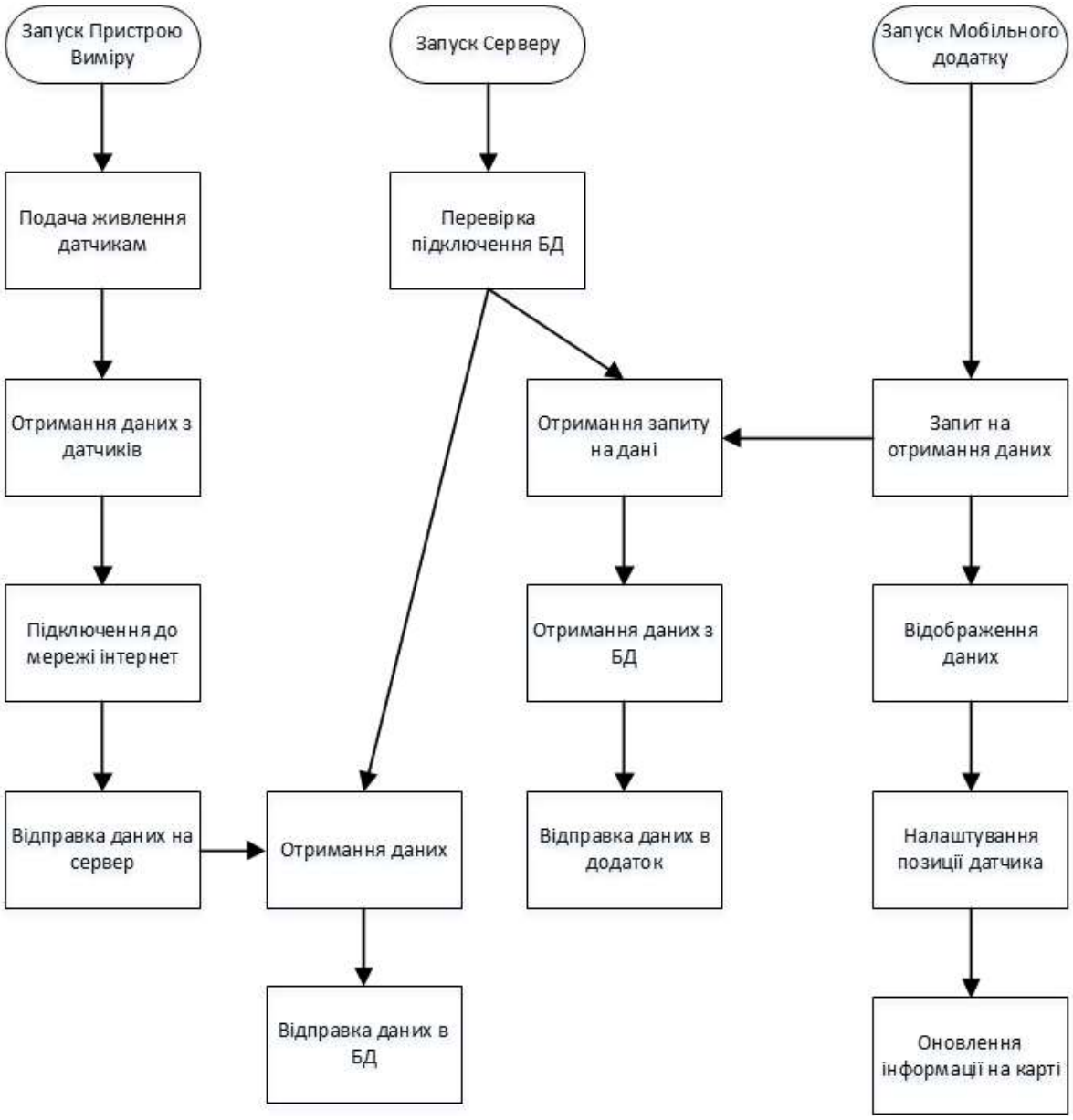
Система для аналізу загазованості повітря для Smart city

Функціональна схема

ІАЛЦ.467200.004 Д1

Листів 1

2020



					ІАЛЦ.467200.004 Д1	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Гайдай А. Р.			Система для аналізу загазованості повітря для Smart city Функціональна схема	Літ. / Арк. 1 / Акрушів 1
Перевір.		Клименко І.А.				КПІ ім. Ігоря Сікорського ФІОТ ІО-63
Н. Контр.		Сімоненко В.П.				
Затверд.		Стіренко С.Г.				

ДОДАТОК 2

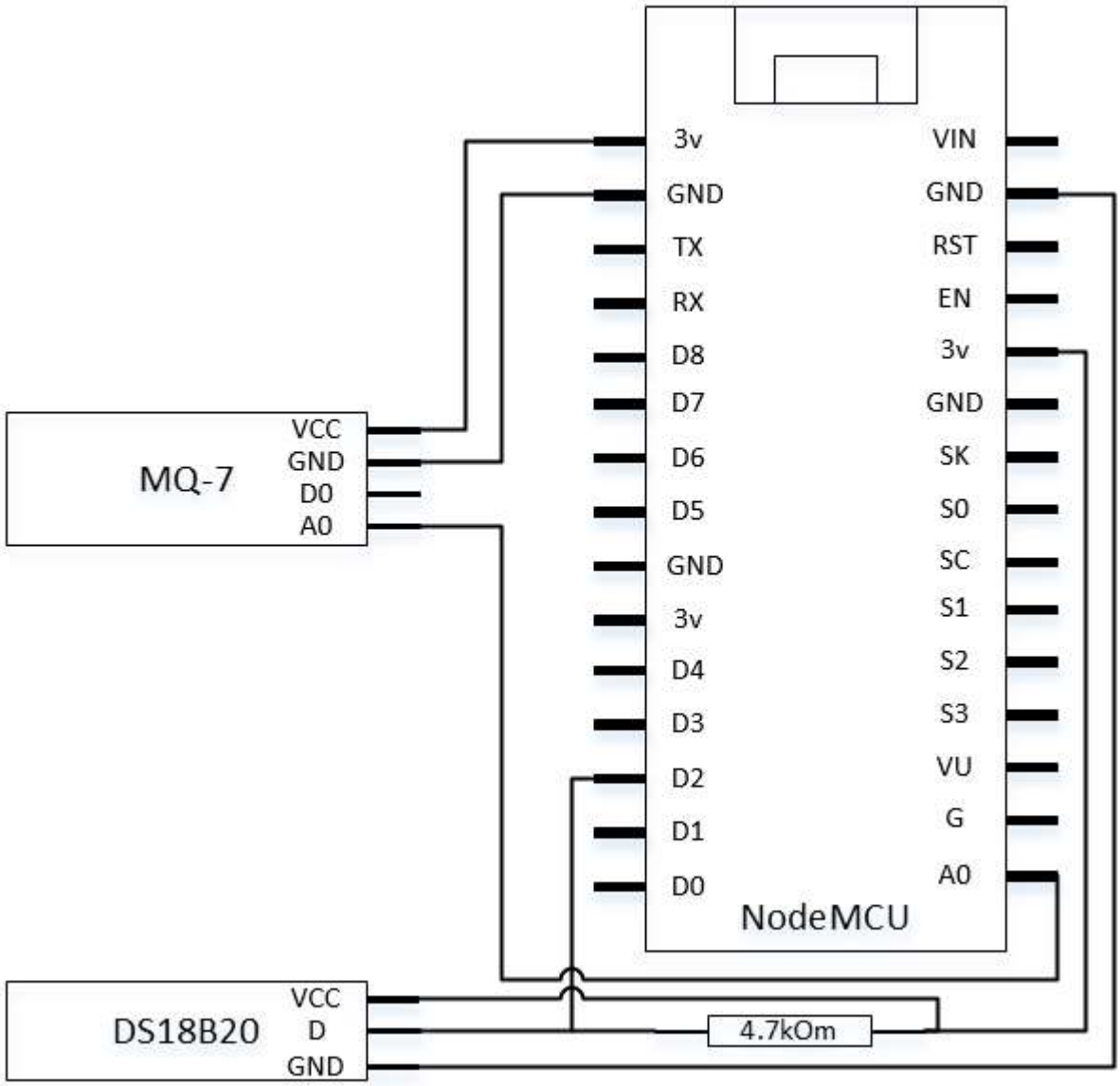
Система для аналізу загазованості повітря для Smart city

Принципова схема

ІАЛЦ.467200.005 Д2

Листів 1

2020



					ІАЛЦ.467200.005 Д2	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Гайдай А. Р.			Система для аналізу загазованості повітря для Smart city Принципова схема	*Літ.* / *Арк.* 1 / *Акрушів* 1
Перевір.		Клименко І.А.				КПІ ім. Ігоря Сікорського ФІОТ ІО-63
Н. Контр.		Сімоненко В.П.				
Затверд.		Стіренко С.Г.				

ДОДАТОК 3

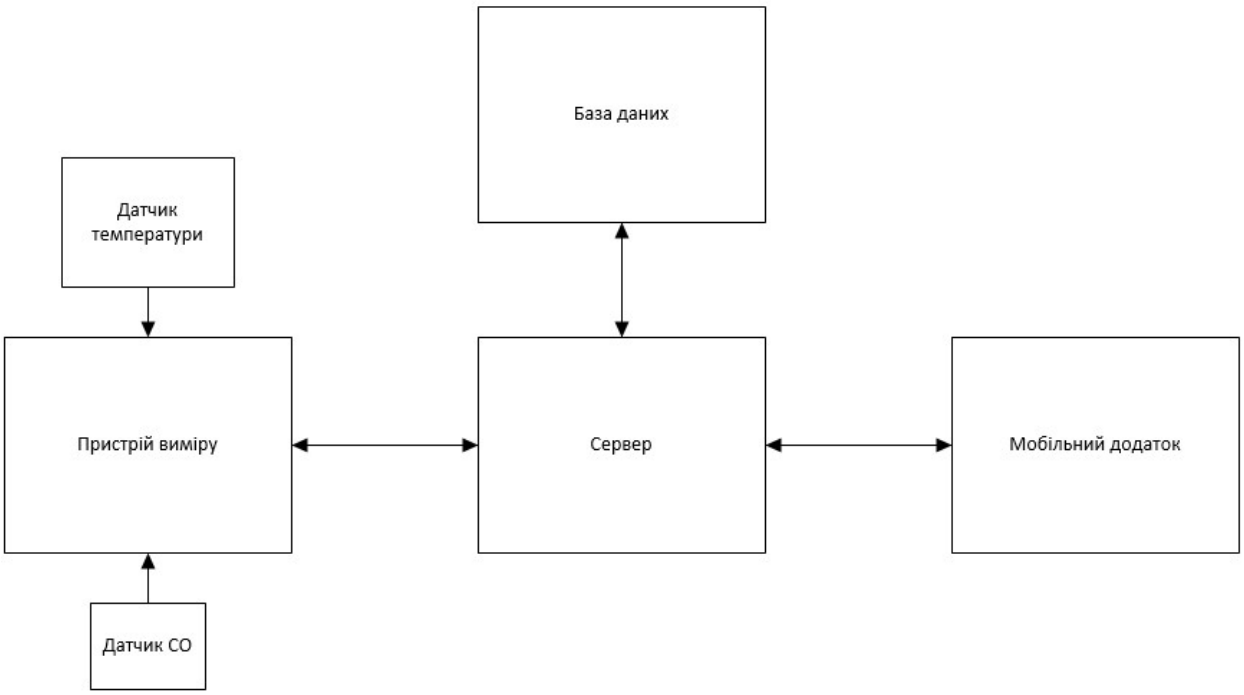
Система для аналізу загазованості повітря для Smart city

Структурна схема

ІАЛЦ.467200.006 Д3

Листів 1

2020



					ІАЛЦ.467200.006 ДЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Гайдай А. Р.			Система для аналізу загазованості повітря для Smart city Структурна схема	Літ.	Арк. 1 / Акрушів 1
Перевір.		Клименко І.А.					
Н. Контр.		Сімоненко В.П.				КПІ ім. Ігоря Сікорського ФІОТ ІО-63	
Затверд.		Стіренко С.Г.					

ДОДАТОК 4

Система для аналізу загазованості повітря для Smart city

Лістинг програми

ІАЛЦ.467200.007 Д4

Листів 10

2020

```
import java.io.*;
import java.net.*;
import java.util.Arrays;

class Server extends Thread {
    private Socket socket;
    private BufferedReader in;
    private BufferedWriter out;

    public Server(Socket socket) throws IOException {
        this.socket = socket;
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
        start();
    }

    @Override
    public void run() {
        String word;
        String[] tokens;
        Connector connector = new Connector();
        try {
            while (true) {
                word = in.readLine();

                if (word != null) {
                    System.out.println(word);
```

```
                tokens = word.split(";");
                try {
                    if (tokens.length == 1) {
                        if (tokens[0].contains("getdata")) {
                            send(connector.dataforMap() + "\n");
                            System.out.println(connector.dataforMap());
                        }
                    }
                    if (tokens.length == 3) {
                        if (CheckInt(tokens[0]) & connector.isUniq(Integer.parseInt(tokens[0]))) {
                            connector.setDataofESP(Integer.parseInt(tokens[0]), false);
                            send("setDataofESP" + "\n");
                        }
                        if (CheckInt(tokens[0])) {//id, co, temp
                            connector.sendData(Integer.parseInt(tokens[0]),  Float.parseFloat(tokens[1]),
Float.parseFloat(tokens[2]));
                            send("sendData" + "\n");
                        }
                    }
                    if (tokens.length == 4) {
                        System.out.println("id, status,length, width");
                        if  (connector.checkSecret(Integer.parseInt(tokens[0])))  {  //  id,  status,length,
width
                            connector.setStatus(Integer.parseInt(tokens[0]), Integer.parseInt(tokens[1]));
                            connector.setGPS(Integer.parseInt(tokens[0]),     Float.parseFloat(tokens[2]),
Float.parseFloat(tokens[3]));
                            send("setGPS" + "\n");
                        }
```

```
                } else {
                    this.downService();
                }
            } catch (NumberFormatException e) {
                e.printStackTrace();
            }
        }
    } catch (IOException e) {
        this.downService();
    }

}

private void send(String msg) {
    try {
        out.write(msg + "\n");
        out.flush();
    } catch (IOException ignored) {
    }
}

private boolean CheckInt(String tokens) {
    boolean check = false;
    try {
        Integer.parseInt(tokens);
        check = true;
```

```
        } catch (NumberFormatException ignored) {
        }
        return check;
    }

    private void downService() {
        try {
            if (!socket.isClosed()) {
                socket.close();
                in.close();
                out.close();
                for (Server vr : Run_Server.serverList) {
                    if (vr.equals(this)) vr.interrupt();
                    Run_Server.serverList.remove(this);
                }
            }
        } catch (IOException ignored) {
        }
    }
}
import java.sql.*;
import java.util.ArrayList;

public class Connector {
    final String myDriver = "com.mysql.jdbc.Driver";
    final String myUrl = "jdbc:mysql://localhost/ESP_Data";
    final String user = "root";
```

```
final String password = "Anatolii";


public void setDataofESP(int ESP_ID, boolean STATUS) {
    try {
        Class.forName(myDriver);
        Connection conn = DriverManager.getConnection(myUrl, user, password);
        String query = " insert into ESP_Connect (unique_id, status)"
                + " values (?, ?)";
        PreparedStatement preparedStmt = conn.prepareStatement(query);
        preparedStmt.setInt(1, ESP_ID);
        preparedStmt.setBoolean(2, STATUS);
        preparedStmt.execute();
        conn.close();
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
}


public boolean isUniq(int ESP_ID) {
    boolean result = true;
    ArrayList<Integer> uniq = new ArrayList<>();
    try {
        Class.forName(myDriver);
        Connection conn = DriverManager.getConnection(myUrl, user, password);


        Statement statement = conn.createStatement();
```

```
        ResultSet resultSet = statement.executeQuery("SELECT DISTINCT unique_id FROM
ESP_Connect");
        while (resultSet.next()) {
            int id = Integer.parseInt(resultSet.getString("unique_id"));
            uniq.add(id);
        }

        conn.close();
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
    for (int id : uniq) {
        if (ESP_ID == id) {
            result = false;
            break;          }       }       return result;    }
  public ArrayList showAllESP() {
    ArrayList<Integer> uniq = new ArrayList<>();
    try {
        Class.forName(myDriver);
        Connection conn = DriverManager.getConnection(myUrl, user, password);

        Statement statement = conn.createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT DISTINCT * FROM
ESP_Connect");
        while (resultSet.next()) {
            int id = Integer.parseInt(resultSet.getString("unique_id"));
            if (resultSet.getInt(2) == 0) {
```

```
                uniq.add(id);
            }
        }
        conn.close();
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
    return uniq;
}

public boolean checkSecret(int id) {
    boolean test = false;
    try {
        Class.forName(myDriver);
        Connection conn = DriverManager.getConnection(myUrl, user, password);

        Statement statement = conn.createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT * FROM ESP_Connect");
        while (resultSet.next()) {
            if (resultSet.getInt("unique_id") == id) {
                test = true;            }        }        conn.close();        } catch (Exception e) {
System.err.println(e.getMessage());        }
    return test;    }
public String dataforMap() {
    StringBuilder test = new StringBuilder();
    try {
        Class.forName(myDriver);
```

```
        Connection conn = DriverManager.getConnection(myUrl, user, password);
        Statement statement = conn.createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT * FROM ESP_data AS ab\n" +
                "WHERE data_time =\n" +
                "  (SELECT MAX(data_time) FROM ESP_data AS ab2 WHERE ab.unique_id = ab2.unique_id)");
        while (resultSet.next()) {
           int id = resultSet.getInt(1);
           float co = resultSet.getFloat(2);
           float temper = resultSet.getFloat(3);
           float length = resultSet.getFloat(4);
           float width = resultSet.getFloat(5);

test.append(id).append(";").append(co).append(";").append(temper).append(";").append(length).append(";").append(width).append(";").append(":");           }


           conn.close();
       } catch (Exception e) {
           System.err.println(e.getMessage());
       }
       return test.toString();
   }
   public void setStatus(int id_un, int stat) {
       try {
           Class.forName(myDriver);
           Connection conn = DriverManager.getConnection(myUrl, user, password);
           PreparedStatement ps = conn.prepareStatement("UPDATE ESP_Connect SET status=" + stat + " WHERE unique_id=" + id_un);
```

```
            ps.executeUpdate();
            ps.close();
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
    public void setGPS(int ESP_ID, float length, float width) {
        try {
            Class.forName(myDriver);
            Connection conn = DriverManager.getConnection(myUrl, user, password);
            Statement statement = conn.createStatement();
            Timestamp search = null;
            ResultSet resultSet = statement.executeQuery("SELECT * FROM ESP_data AS ab\n" +
                    "WHERE data_time =\n" +
                    "  (SELECT MAX(data_time) FROM ESP_data AS ab2 WHERE ab.unique_id = ab2.unique_id) AND unique_id =" + ESP_ID);
            while (resultSet.next()) {
               search = resultSet.getTimestamp(6);           }
            PreparedStatement ps = conn.prepareStatement("UPDATE ESP_data SET lenth=" + length + " , width=" + width +
                    " WHERE unique_id=" + ESP_ID + " AND data_time='" + search + "'");
            ps.executeUpdate();
            ps.close();
            conn.close();
        } catch (Exception e) {
            System.err.println(e.getMessage());}}
    public void sendData(int ESP_ID, float CO, float TEMPERATURA) {
```

```
try {
    float l = 0;
    float w = 0;
    Class.forName(myDriver);
    Connection conn = DriverManager.getConnection(myUrl, user, password);
    Statement statement = conn.createStatement();
    ResultSet resultSet = statement.executeQuery("SELECT * FROM ESP_data AS ab\n" +
        "WHERE data_time =\n" +
        "  (SELECT MAX(data_time) FROM ESP_data AS ab2 WHERE ab.unique_id = ab2.unique_id) AND unique_id =" + ESP_ID);
    while (resultSet.next()) {
      l = resultSet.getFloat(4);
      w = resultSet.getFloat(5);          }
    String query = "insert into ESP_data(unique_id, CO, temp_data, lenth, width, data_time)"
        + " values (?, ?, ?, ?, ?, ?)";
    PreparedStatement preparedStmt = conn.prepareStatement(query);
    preparedStmt.setInt(1, ESP_ID);
    preparedStmt.setFloat(2, CO);
    preparedStmt.setFloat(3, TEMPERATURA);
    preparedStmt.setFloat(4, l);
    preparedStmt.setFloat(5, w);
    preparedStmt.setTimestamp(6, new Timestamp(System.currentTimeMillis()));
    preparedStmt.execute();
    conn.close();
} catch (Exception e) {
    System.err.println(e.getMessage());}}}
```